NP-completeness - Part II

K. Subramani

1Lane Department of Computer Science and Electrical Engineering
West Virginia University

April 6, 2015
Outline

1. Optimization Problems on Graphs
Outline

1. Optimization Problems on Graphs
2. Number Problems
Outline

1. Optimization Problems on Graphs
2. Number Problems
3. The Power of Integer Programming
Outline

1. Optimization Problems on Graphs
2. Number Problems
3. The Power of Integer Programming
4. Paths, trees and Circuits
Optimization Problems on Graphs
Number Problems
The Power of Integer Programming
Paths, trees and Circuits

Independent Set

Definition
Input: An undirected graph $G = \langle V, E \rangle$ and a number $K \leq |V|$.
Query: Is there a set $V' \subseteq V$ with $|V'| \geq K$ such that for any two vertices $u, v \in V'$, $(u, v) \not\in E$?

NP-completeness
Computational Complexity
Independent Set

Definition

Input:

An undirected graph $G = \langle V, E \rangle$ and a number $K \leq |V|$.

Query:

Is there a set $V' \subseteq V$, with $|V'| \geq K$ such that for any two vertices $u, v \in V'$, $(u, v) \not\in E$?

NP-completeness

Computational Complexity
Independent Set

**Definition**

**Input:** An undirected graph $G = \langle V, E \rangle$ and a number $K \leq |V|$. 

**NP-completeness**

Computational Complexity
Independent Set

**Definition**

**Input:** An undirected graph \( G = \langle V, E \rangle \) and a number \( K \leq |V| \).

**Query:** Is there a set \( V' \subseteq V \), with \( |V'| \geq K \) such that for any two vertices \( u, v \in V' \), \((u, v) \not\in E\)?
Example

In the graph below, \( V' = \{v_2, v_4\} \) is an independent set.
In the graph below, \( V' = \{ v_2, v_4 \} \) is an independent set.
Example

In the graph below, $V' = \{v_2, v_4\}$ is an independent set.
Example

In the graph below, $V' = \{v_2, v_4\}$ is an independent set.
Theorem

**INDEPENDENT-SET is NP-complete.**
Theorem

**INDEPENDENT-SET** is **NP-complete**.

Proof

1. **INDEPENDENT-SET** is clearly in **NP**.
2. We reduce 3SAT to **INDEPENDENT-SET**.
3. Given an instance $\phi$ of 3SAT with $m$ clauses and $n$ variables, we construct a graph $G = \langle V, E \rangle$ as follows:
   - For each one of the $m$ clauses, we create a separate triangle in the graph.
   - Each node of the triangle corresponds to a literal in the clause.
   - There is an edge between two nodes $u$ and $v$ in different triangles if and only if $v = \neg u$.
4. Set $K = m$. 

NP-completeness

Computational Complexity
Theorem

INDEPENDENT-SET is NP-complete.

Proof

1. INDEPENDENT-SET is clearly in NP.
**Theorem**

INDEPENDENT-SET is NP-complete.

**Proof**

1. INDEPENDENT-SET is clearly in NP.
2. We reduce 3SAT to INDEPENDENT-SET.
Theorem

**INDEPENDENT-SET** is *NP-complete*.

Proof

1. **INDEPENDENT-SET** is clearly in **NP**.
2. We reduce 3SAT to **INDEPENDENT-SET**.
3. Given an instance \( \phi \) of 3SAT with \( m \) clauses and \( n \) variables,
**Theorem**

**INDEPENDENT-SET** is NP-complete.

**Proof**

1. **INDEPENDENT-SET** is clearly in NP.
2. We reduce 3SAT to **INDEPENDENT-SET**.
3. Given an instance $\phi$ of 3SAT with $m$ clauses and $n$ variables, we construct a graph $G = \langle V, E \rangle$ as follows:
Theorem

**INDEPENDENT-SET** is **NP-complete**.

Proof

1. **INDEPENDENT-SET** is clearly in **NP**.
2. We reduce **3SAT** to **INDEPENDENT-SET**.
3. Given an instance $\phi$ of **3SAT** with $m$ clauses and $n$ variables, we construct a graph $G = \langle V, E \rangle$ as follows:
   - For each one of the $m$ clauses, we create a separate triangle in the graph.
Theorem

**INDEPENDENT-SET** is **NP-complete**.

Proof

1. **INDEPENDENT-SET** is clearly in **NP**.
2. We reduce 3SAT to **INDEPENDENT-SET**.
3. Given an instance \( \phi \) of 3SAT with \( m \) clauses and \( n \) variables, we construct a graph \( G = \langle V, E \rangle \) as follows:
   - For each one of the \( m \) clauses, we create a separate triangle in the graph.
   - Each node of the triangle corresponds to a literal in the clause.
Theorem

INDEPENDENT-SET is **NP-complete**.

Proof

1. **INDEPENDENT-SET** is clearly in **NP**.
2. We reduce 3SAT to **INDEPENDENT-SET**.
3. Given an instance \( \phi \) of 3SAT with \( m \) clauses and \( n \) variables, we construct a graph \( G = \langle V, E \rangle \) as follows:
   - For each one of the \( m \) clauses, we create a separate triangle in the graph.
   - Each node of the triangle corresponds to a literal in the clause.
   - There is an edge between two nodes \( u \) and \( v \) in different triangles if and only if \( v = \neg u \).
Theorem

**INDEPENDENT-SET** is **NP-complete**.

Proof

1. INDEPENDENT-SET is clearly in **NP**.
2. We reduce 3SAT to INDEPENDENT-SET.
3. Given an instance $\phi$ of 3SAT with $m$ clauses and $n$ variables, we construct a graph $G = \langle V, E \rangle$ as follows:
   - For each one of the $m$ clauses, we create a separate triangle in the graph.
   - Each node of the triangle corresponds to a literal in the clause.
   - There is an edge between two nodes $u$ and $v$ in different triangles if and only if $v = \neg u$.
   - Set $K = m$. 

**NP-completeness**

Computational Complexity
Graphical representation

Example

\(\phi = (x_1 \lor x_2 \lor x_3) \land (\neg x_1 \lor \neg x_2 \lor \neg x_3) \land (\neg x_1 \lor x_2 \lor x_3)\)

NP-completeness

Computational Complexity
Graphical representation

Example

\[ \phi = (x_1 \lor x_2 \lor x_3) \land (\neg x_1 \lor \neg x_2 \lor \neg x_3) \land (\neg x_1 \lor x_2 \lor x_3) \]
Example

\[ \phi = (x_1 \lor x_2 \lor x_3) \land (\neg x_1 \lor \neg x_2 \lor \neg x_3) \land (\neg x_1 \lor x_2 \lor x_3) \]

Graphical representation
Completing the Reduction

Proof

We claim that $\phi$ is satisfiable if and only if there is an independent set $V'$ of $K$ nodes in graph $R(\phi)$.

1. Assume that a satisfying assignment exists for $\phi$.
2. Pick a node in each clause triangle that is set to true under this assignment.
3. The set of picked nodes must be independent. Why?
4. We thus have an independent set of size $\geq K = m$.
5. Now, assume that we have an independent set $V'$ in $R(\phi)$ such that $|V'| \geq m$.
6. Then, $|V'| = m$. Why?
7. Set the literal corresponding to the vertex picked from each triangle to true.
8. Since no pair of complementary literals is picked, the truth assignment is consistent.
9. One literal from each clause is set to true and hence all clauses are satisfied.
We claim that $\phi$ is satisfiable if and only if there is an independent set $V'$ of $K$ nodes in graph $R(\phi)$.

1. Assume that a satisfying assignment exists for $\phi$.
2. Pick a node in each clause triangle that is set to true under this assignment.
3. The set of picked nodes must be independent. Why?
4. We thus have an independent set of size $\geq K = m$.
5. Now, assume that we have an independent set $V'$ in $R(\phi)$ such that $|V'| \geq m$.
6. Then, $|V'| = m$. Why?
7. Set the literal corresponding to the vertex picked from each triangle to true.
8. Since no pair of complementary literals is picked, the truth assignment is consistent.
9. One literal from each clause is set to true and hence all clauses are satisfied.
Completing the Reduction

Proof

We claim that $\phi$ is satisfiable if and only if there is an independent set $V'$ of $K$ nodes in graph $R(\phi)$.
We claim that $\phi$ is satisfiable if and only if there is an independent set $V'$ of $K$ nodes in graph $R(\phi)$.

1. Assume that a satisfying assignment exists for $\phi$. 

Proof
Proof

We claim that $\phi$ is satisfiable if and only if there is an independent set $V'$ of $K$ nodes in graph $R(\phi)$.

1. Assume that a satisfying assignment exists for $\phi$.
2. Pick a node in each clause triangle that is set to true under this assignment.
Completing the Reduction

Proof

We claim that $\phi$ is satisfiable if and only if there is an independent set $V'$ of $K$ nodes in graph $R(\phi)$.

1. Assume that a satisfying assignment exists for $\phi$.
2. Pick a node in each clause triangle that is set to true under this assignment.
3. The set of picked nodes must be independent. Why?

Now, assume that we have an independent set $V'$ in $R(\phi)$ such that $|V'| \geq m$.

Then, $|V'| = m$.

Set the literal corresponding to the vertex picked from each triangle to true.

Since no pair of complementary literals is picked, the truth assignment is consistent.

One literal from each clause is set to true and hence all clauses are satisfied.
Proof

We claim that $\phi$ is satisfiable if and only if there is an independent set $V'$ of $K$ nodes in graph $R(\phi)$.

1. Assume that a satisfying assignment exists for $\phi$.
2. Pick a node in each clause triangle that is set to true under this assignment.
3. The set of picked nodes must be independent. Why?
4. We thus have an independent set of size $\geq K = m$. 

Completing the Reduction

Proof

We claim that $\phi$ is satisfiable if and only if there is an independent set $V'$ of $K$ nodes in graph $R(\phi)$.

1. Assume that a satisfying assignment exists for $\phi$.
2. Pick a node in each clause triangle that is set to true under this assignment.
3. The set of picked nodes must be independent. Why?
4. We thus have an independent set of size $\geq K = m$.
5. Now, assume that we have an independent set $V'$ in $R(\phi)$ such that $|V'| \geq m$. 

Proof

We claim that $\phi$ is satisfiable if and only if there is an independent set $V'$ of $K$ nodes in graph $R(\phi)$.

1. Assume that a satisfying assignment exists for $\phi$.
2. Pick a node in each clause triangle that is set to true under this assignment.
3. The set of picked nodes must be independent. Why?
4. We thus have an independent set of size $\geq K = m$.
5. Now, assume that we have an independent set $V'$ in $R(\phi)$ such that $|V'| \geq m$.
6. Then, $|V'| = m$. 
We claim that $\phi$ is satisfiable if and only if there is an independent set $V'$ of $K$ nodes in graph $R(\phi)$.

1. Assume that a satisfying assignment exists for $\phi$.
2. Pick a node in each clause triangle that is set to true under this assignment.
3. The set of picked nodes must be independent. Why?
4. We thus have an independent set of size $\geq K = m$.
5. Now, assume that we have an independent set $V'$ in $R(\phi)$ such that $|V'| \geq m$.
6. Then, $|V'| = m$. Why?
Proof

We claim that $\phi$ is satisfiable if and only if there is an independent set $V'$ of $K$ nodes in graph $R(\phi)$.

1. Assume that a satisfying assignment exists for $\phi$.
2. Pick a node in each clause triangle that is set to true under this assignment.
3. The set of picked nodes must be independent. Why?
4. We thus have an independent set of size $\geq K = m$.
5. Now, assume that we have an independent set $V'$ in $R(\phi)$ such that $|V'| \geq m$.
6. Then, $|V'| = m$. Why?
7. Set the literal corresponding to the vertex picked from each triangle to true.
Proof

We claim that $\phi$ is satisfiable if and only if there is an independent set $V'$ of $K$ nodes in graph $R(\phi)$.

1. Assume that a satisfying assignment exists for $\phi$.
2. Pick a node in each clause triangle that is set to true under this assignment.
3. The set of picked nodes must be independent. Why?
4. We thus have an independent set of size $\geq K = m$.
5. Now, assume that we have an independent set $V'$ in $R(\phi)$ such that $|V'| \geq m$.
6. Then, $|V'| = m$. Why?
7. Set the literal corresponding to the vertex picked from each triangle to true.
8. Since no pair of complementary literals is picked, the truth assignment is consistent.
Proof

We claim that $\phi$ is satisfiable if and only if there is an independent set $V'$ of $K$ nodes in graph $R(\phi)$.

1. Assume that a satisfying assignment exists for $\phi$.
2. Pick a node in each clause triangle that is set to true under this assignment.
3. The set of picked nodes must be independent. Why?
4. We thus have an independent set of size $\geq K = m$.
5. Now, assume that we have an independent set $V'$ in $R(\phi)$ such that $|V'| \geq m$.
6. Then, $|V'| = m$. Why?
7. Set the literal corresponding to the vertex picked from each triangle to true.
8. Since no pair of complementary literals is picked, the truth assignment is consistent.
9. One literal from each clause is set to true and hence all clauses are satisfied.
Vertex-Cover

Definition

Input: An undirected graph $G = \langle V, E \rangle$ and a number $K \leq |V|$.

Query: Is there a set $V' \subseteq V$, with $|V'| \leq K$ such that for any two vertices $u, v \in V$, $(u, v) \in E \rightarrow (u \in V' \lor v \in V')$?
**Definition**

**Input:** An undirected graph $G = \langle V, E \rangle$ and a number $K \leq |V|$.

**Query:** Is there a set $V' \subseteq V$, with $|V'| \leq K$ such that for any two vertices $u, v \in V$, $(u, v) \in E \rightarrow (u \in V' \lor v \in V')$?
Definition

**Input:** An undirected graph $G = \langle V, E \rangle$ and a number $K \leq |V|$. 
Definition

**Input:** An undirected graph $G = (V, E)$ and a number $K \leq |V|$.

**Query:** Is there a set $V' \subseteq V$, with $|V'| \leq K$ such that for any two vertices $u, v \in V, (u, v) \in E$.
**Definition**

**Input:** An undirected graph $G = \langle V, E \rangle$ and a number $K \leq |V|$.

**Query:** Is there a set $V' \subseteq V$, with $|V'| \leq K$ such that for any two vertices $u, v \in V$, $(u, v) \in E \rightarrow (u \in V')$ or $v \in V'$?
Theorem

\[ \text{VERTEX-COVER} \text{ is NP-complete.} \]

Proof

1. \( \text{VERTEX-COVER} \) is clearly in \( \text{NP} \).
2. We reduce \( \text{INDEPENDENT-SET} \) to \( \text{VERTEX-COVER} \).
3. Let \((G = \langle V, E \rangle, K)\) denote an instance of the \( \text{INDEPENDENT-SET} \) problem.
4. The corresponding instance of the \( \text{VERTEX-COVER} \) problem is \((G = \langle V, E \rangle, |V| - K)\).
5. The crucial observation is that the vertex complement of a covering set must be independent and vice versa.
Theorem

**Vertex-Cover is NP-complete.**
Theorem

**VERTEX-COVER** is **NP-complete**.

Proof

1. **VERTEX-COVER** is clearly in **NP**.
2. We reduce **INDEPENDENT-SET** to **VERTEX-COVER**.
3. Let $(G = \langle V, E \rangle, K)$ denote an instance of the **INDEPENDENT-SET** problem.
4. The corresponding instance of the **VERTEX-COVER** problem is $(G = \langle V, E \rangle, |V| - K)$.
5. The crucial observation is that the vertex complement of a covering set must be independent and vice versa.
**Theorem**

Vertex-Cover is **NP-complete**.

**Proof**

1. Vertex-Cover is clearly in NP.
Vertex-Cover is NP-complete.

Proof:

1. Vertex-Cover is clearly in NP.
2. We reduce Independent-Set to Vertex-Cover.
Theorem

**VERTEX-COVER** is **NP-complete**.

Proof

1. **VERTEX-COVER** is clearly in **NP**.
2. We reduce **INDEPENDENT-SET** to **VERTEX-COVER**.
3. Let \((G = \langle V, E \rangle, K)\) denote an instance of the **INDEPENDENT-SET** problem.
Theorem

**VERTEX-COVER is NP-complete.**

Proof

1. **VERTEX-COVER** is clearly in **NP**.
2. We reduce **INDEPENDENT-SET** to **VERTEX-COVER**.
3. Let \((G = \langle V, E \rangle, K)\) denote an instance of the **INDEPENDENT-SET** problem.
4. The corresponding instance of the **VERTEX-COVER** problem is \((G = \langle V, E \rangle, |V| - K)\).
Theorem

\textbf{VERTEX-COVER} is \textbf{NP-complete}.

Proof

1. \textbf{VERTEX-COVER} is clearly in \textbf{NP}.
2. We reduce \textbf{INDEPENDENT-SET} to \textbf{VERTEX-COVER}.
3. Let \((G = \langle V, E \rangle, K)\) denote an instance of the \textbf{INDEPENDENT-SET} problem.
4. The corresponding instance of the \textbf{VERTEX-COVER} problem is \((G = \langle V, E \rangle, |V| - K)\).
5. The crucial observation is that the vertex complement of a covering set must be independent and vice versa.
Clique

Definition

Input:
An undirected graph $G = \langle V, E \rangle$ and a number $K \leq |V|$. 

Query:
Is there a set $V' \subseteq V$, with $|V'| \geq K$ such that for any two vertices $u, v \in V'$, $(u, v) \in E$?
Optimization Problems on Graphs
Number Problems
The Power of Integer Programming
Paths, trees and Circuits

Clique

Definition

Input: An undirected graph \( G = \langle V, E \rangle \) and a number \( K \leq |V| \).
Query: Is there a set \( V' \subseteq V \) with \( |V'| \geq K \) such that for any two vertices \( u, v \in V' \), \((u, v) \in E\)?
Definition

**Input:** An undirected graph $G = \langle V, E \rangle$ and a number $K \leq |V|$. 
Clique

Definition

Input: An undirected graph $G = \langle V, E \rangle$ and a number $K \leq |V|$.

Query: Is there a set $V' \subseteq V$, with $|V'| \geq K$ such that for any two vertices $u, v \in V'$, $(u, v) \in E$?
Theorem

CLIQUE is NP-complete.

Proof
1. CLIQUE is clearly in NP.
2. We reduce INDEPENDENT-SET to CLIQUE.
3. Let \((G = \langle V, E \rangle, K)\) denote an instance of the INDEPENDENT-SET problem.
4. The corresponding instance of the CLIQUE problem is \((G_c = \langle V, E_c \rangle, K)\).
5. The crucial observation is that any independent set in \(G\) corresponds to a clique of the same size in \(G_c\) and vice versa.
Theorem

**Clique** is NP-complete.

**Proof**

1. **Clique** is clearly in **NP**.
2. We reduce **INDEPENDENT-SET** to **Clique**.
3. Let \((G = \langle V, E \rangle, K)\) denote an instance of the **INDEPENDENT-SET** problem.
4. The corresponding instance of the **Clique** problem is \((G_c = \langle V, E_c \rangle, K)\).
5. The crucial observation is that any independent set in \(G\) corresponds to a clique of the same size in \(G_c\) and vice versa.
Theorem

\textbf{Clique is NP-complete.}
Theorem

\textbf{Clique} is \textbf{NP-complete}.

Proof
**Theorem**

**CLIQUE** is **NP-complete**.

**Proof**

1. **CLIQUE** is clearly in **NP**.
Theorem

Clique is NP-complete.

Proof

1. Clique is clearly in NP.
2. We reduce Independent-Set to Clique.
Theorem

**CLIQUE** is **NP-complete**.

Proof

1. **CLIQUE** is clearly in **NP**.
2. We reduce **INDEPENDENT-SET** to **CLIQUE**.
3. Let \((G = \langle V, E \rangle, K)\) denote an instance of the **INDEPENDENT-SET** problem.
**Theorem**

*Clique* is **NP-complete**.

**Proof**

1. *Clique* is clearly in **NP**.
2. We reduce **Independent-Set** to *Clique*.
3. Let \((G = \langle V, E \rangle, K)\) denote an instance of the **Independent-Set** problem.
4. The corresponding instance of the *Clique* problem is \((G^c = \langle V, E^c \rangle, K)\).
**Theorem**

CLIQUE is **NP-complete**.

**Proof**

1. CLIQUE is clearly in NP.
2. We reduce INDEPENDENT-SET to CLIQUE.
3. Let \((G = \langle V, E \rangle, K)\) denote an instance of the INDEPENDENT-SET problem.
4. The corresponding instance of the CLIQUE problem is \((G^c = \langle V, E^c \rangle, K)\).
5. The crucial observation is that any independent set in \(G\) corresponds to a clique of the same size in \(G^c\) and vice versa.
Graph 3-Colorability

Definition

Input: An undirected graph $G = \langle V, E \rangle$ and a set $C = \{0, 1, 2\}$.

Query: Is there a function $f : V \rightarrow C$, such that for all $(u, v) \in E$, $f(u) \neq f(v)$?
Graph 3-Colorability

**Definition**

Input: An undirected graph \( G = \langle V, E \rangle \) and a set \( C = \{0, 1, 2\} \).

Query: Is there a function \( f: V \rightarrow C \), such that for all \((u, v) \in E\), \( f(u) \neq f(v) \)?

**NP-completeness**
Graph 3-Colorability

Definition

**Input:** An undirected graph $G = \langle V, E \rangle$ and a set $C = \{0, 1, 2\}$. 
Graph 3-Colorability

Definition

**Input:** An undirected graph $G = \langle V, E \rangle$ and a set $C = \{0, 1, 2\}$.

**Query:** Is there a function $f : V \rightarrow C$, such that for all $(u, v) \in E$, $f(u) \neq f(v)$?
Theorem

GRAPH 3-COLORABILITY is NP-complete.

Proof

1. GRAPH 3-COLORABILITY is clearly in NP.

2. We reduce NAE3SAT to GRAPH 3-COLORABILITY.

3. Let $\phi = C_1 \land C_2 \ldots C_m$ be a 3CNF formula over $n$ variables and $m$ clauses.

4. The corresponding instance of GRAPH 3-COLORABILITY is the graph $G = \langle V, E \rangle$ constructed as follows:

   1. $V = \{a\} \cup \{x_i, \neg x_i\}, \forall i = 1, 2, \ldots, n \cup \{C_{i1}, C_{i2}, C_{i3}\}, \forall i = 1, 2, \ldots, m$,

   2. $E_1 = \{a, x_i\}, \forall i = 1, 2, \ldots, n \cup \{a, \neg x_i\}, \forall i = 1, 2, \ldots, n$.

   3. $E_2 = \{C_{i1}, C_{i2}\} \cup \{C_{i1}, C_{i3}\} \cup \{C_{i2}, C_{i3}\}, \forall i = 1, 2, \ldots, m$.

   4. $E_3 = \bigcup \{C_{ij}, x_k\}, \forall j = 1, 2, 3, \forall i = 1, 2, \ldots, m, \forall k = 1, 2, \ldots, n$, if $C_{ij} = x_k$.

   5. $E_4 = \bigcup \{C_{ij}, \neg x_k\}, \forall j = 1, 2, 3, \forall i = 1, 2, \ldots, m, \forall k = 1, 2, \ldots, n$, if $C_{ij} = \neg x_k$.

   6. $E_5 = \bigcup \{x_i, \neg x_i\}, \forall i = 1, 2, \ldots, n$.

   7. $E = E_1 \cup E_2 \cup E_3 \cup E_4 \cup E_5$. 
Theorem

**GRAPH 3-COLORABILITY** *is NP-complete.*
**Theorem**

**GRAPH 3-COLORABILITY** is **NP-complete**.

**Proof**
GRAPH 3-COLORABILITY is NP-complete.

Proof

1. GRAPH 3-COLORABILITY is clearly in NP.
Theorem

**GRAPH 3-COLORABILITY** is **NP-complete**.

Proof

1. **GRAPH 3-COLORABILITY** is clearly in **NP**.
2. We reduce NAE3SAT to **GRAPH 3-COLORABILITY**.
Theorem

**GRAPH 3-COLORABILITY** is **NP-complete**.

Proof

1. **GRAPH 3-COLORABILITY** is clearly in **NP**.
2. We reduce NAE3SAT to **GRAPH 3-COLORABILITY**.
3. Let \( \phi = C_1 \land C_2 \ldots C_m \) be a 3CNF formula over \( n \) variables and \( m \) clauses.
Theorem

**GRAPH 3-COLORABILITY is NP-complete.**

Proof

1. **GRAPH 3-COLORABILITY** is clearly in **NP**.
2. We reduce NAE3SAT to **GRAPH 3-COLORABILITY**.
3. Let \( \phi = C_1 \land C_2 \ldots C_m \) be a 3CNF formula over \( n \) variables and \( m \) clauses.
4. The corresponding instance of **GRAPH 3-COLORABILITY** is the graph \( G = \langle V, E \rangle \) constructed as follows:
Theorem

**GRAPH 3-COLORABILITY** is **NP-complete**.

Proof

1. **GRAPH 3-COLORABILITY** is clearly in **NP**.
2. We reduce NAE3SAT to **GRAPH 3-COLORABILITY**.
3. Let \( \phi = C_1 \land C_2 \ldots C_m \) be a 3CNF formula over \( n \) variables and \( m \) clauses.
4. The corresponding instance of **GRAPH 3-COLORABILITY** is the graph \( G = \langle V, E \rangle \) constructed as follows:
   1. \( V = \{ a \} \)
Theorem

**GRAPH 3-COLORABILITY** is **NP-complete**.

Proof

1. **GRAPH 3-COLORABILITY** is clearly in **NP**.
2. We reduce NAE3SAT to **GRAPH 3-COLORABILITY**.
3. Let $\phi = C_1 \land C_2 \ldots C_m$ be a 3CNF formula over $n$ variables and $m$ clauses.
4. The corresponding instance of **GRAPH 3-COLORABILITY** is the graph $G = \langle V, E \rangle$ constructed as follows:
   1. $V = \{a\} \cup \{x_i, \neg x_i\}, \forall i = 1, 2, \ldots, n$
**Theorem**

**GRAPH 3-COLORABILITY** is **NP-complete**.

**Proof**

1. **GRAPH 3-COLORABILITY** is clearly in **NP**.
2. We reduce **NAE3SAT** to **GRAPH 3-COLORABILITY**.
3. Let $\phi = C_1 \land C_2 \ldots C_m$ be a 3CNF formula over $n$ variables and $m$ clauses.
4. The corresponding instance of **GRAPH 3-COLORABILITY** is the graph $G = \langle V, E \rangle$ constructed as follows:
   1. $V = \{a\} \cup \{x_i, \neg x_i\}, \forall i = 1, 2, \ldots, n \cup \{C_{i1}, C_{i2}, C_{i3}\}, \forall i = 1, 2 \ldots m$, where $C_{ij}$ refers to the $j^{th}$ literal in the clause $C_i$. 
   2. $E_1 = \{a, x_i\}, \forall i = 1, 2, \ldots, n \cup \{a, \neg x_i\}, \forall i = 1, 2, \ldots, n$.
   3. $E_2 = \{C_{i1}, C_{i2}\} \cup \{C_{i1}, C_{i3}\} \cup \{C_{i2}, C_{i3}\}, \forall i = 1, 2 \ldots m$.
   4. $E_3 = \{x_{ik}, \neg x_{ik}\}, \forall j = 1, 2, 3, k = 1, 2, \ldots n, \forall i = 1, 2 \ldots m$.
   5. $E = E_1 \cup E_2 \cup E_3$. 

Theorem

**GRAPH 3-COLORABILITY** is **NP-complete**.

Proof

1. **GRAPH 3-COLORABILITY** is clearly in **NP**.
2. We reduce NAE3SAT to **GRAPH 3-COLORABILITY**.
3. Let $\phi = C_1 \land C_2 \ldots C_m$ be a 3CNF formula over $n$ variables and $m$ clauses.
4. The corresponding instance of **GRAPH 3-COLORABILITY** is the graph $G = \langle V, E \rangle$ constructed as follows:
   1. $V = \{a\} \cup \{x_i, \neg x_i\}, \forall i = 1, 2, \ldots, n \cup \{C_{i1}, C_{i2}, C_{i3}\}, \forall i = 1, 2 \ldots m$, where $C_{ij}$ refers to the $j^{th}$ literal in the clause $C_i$.
   2. $E_1 = \{a, x_i\}, \forall i = 1, 2, \ldots n$.
Theorem

**Theorem**

**GRAPH 3-COLORABILITY** is **NP-complete**.

Proof

1. **GRAPH 3-COLORABILITY** is clearly in **NP**.
2. We reduce NAE3SAT to **GRAPH 3-COLORABILITY**.
3. Let $\phi = C_1 \land C_2 \ldots C_m$ be a 3CNF formula over $n$ variables and $m$ clauses.
4. The corresponding instance of **GRAPH 3-COLORABILITY** is the graph $G = \langle V, E \rangle$ constructed as follows:
   - $V = \{a\} \cup \{x_i, \neg x_i\}, \forall i = 1, 2, \ldots, n \cup \{C_{i1}, C_{i2}, C_{i3}\}, \forall i = 1, 2 \ldots m$,
   - where $C_{ij}$ refers to the $j^{th}$ literal in the clause $C_i$.
   - $E_1 = \{a, x_i\}, \forall i = 1, 2, \ldots n \cup \{a, \neg x_i\}, \forall i = 1, 2, \ldots n$. 

NP-completeness  Computational Complexity
**Theorem**

**GRAPH 3-COLORABILITY** is **NP-complete**.

**Proof**

1. **GRAPH 3-COLORABILITY** is clearly in **NP**.
2. We reduce NAE3SAT to **GRAPH 3-COLORABILITY**.
3. Let \( \phi = C_1 \land C_2 \ldots C_m \) be a 3CNF formula over \( n \) variables and \( m \) clauses.
4. The corresponding instance of **GRAPH 3-COLORABILITY** is the graph \( G = \langle V, E \rangle \) constructed as follows:
   1. \( V = \{a\} \cup \{x_i, \neg x_i\}, \forall i = 1, 2, \ldots, n \cup \{C_{i1}, C_{i2}, C_{i3}\}, \forall i = 1, 2 \ldots m \), where \( C_{ij} \) refers to the \( j^{th} \) literal in the clause \( C_i \).
   2. \( E_1 = \{a, x_i\}, \forall i = 1, 2, \ldots n \cup \{a, \neg x_i\}, \forall i = 1, 2, \ldots n \).
   3. \( E_2 = \{C_{i1}, C_{i2}\} \).
**Theorem**

GRAPH 3-COLORABILITY is **NP-complete**.

**Proof**

1. **GRAPH 3-COLORABILITY** is clearly in **NP**.
2. We reduce NAE3SAT to **GRAPH 3-COLORABILITY**.
3. Let $\phi = C_1 \land C_2 \ldots C_m$ be a 3CNF formula over $n$ variables and $m$ clauses.
4. The corresponding instance of **GRAPH 3-COLORABILITY** is the graph $G = (V, E)$ constructed as follows:
   1. $V = \{a\} \cup \{x_i, \neg x_i\}, \forall i = 1, 2, \ldots, n \cup \{C_{i1}, C_{i2}, C_{i3}\}, \forall i = 1, 2 \ldots m$,
      where $C_{ij}$ refers to the $j^{th}$ literal in the clause $C_i$.
   2. $E_1 = \{a, x_i\}, \forall i = 1, 2, \ldots n \cup \{a, \neg x_i\}, \forall i = 1, 2, \ldots n$.
   3. $E_2 = \{C_{i1}, C_{i2}\} \cup \{C_{i1}, C_{i3}\}$
Theorem

**GRAPH 3-COLORABILITY** is **NP-complete**.

Proof

1. **GRAPH 3-COLORABILITY** is clearly in **NP**.
2. We reduce **NAE3SAT** to **GRAPH 3-COLORABILITY**.
3. Let $\phi = C_1 \land C_2 \ldots C_m$ be a 3CNF formula over $n$ variables and $m$ clauses.
4. The corresponding instance of **GRAPH 3-COLORABILITY** is the graph $G = \langle V, E \rangle$ constructed as follows:
   1. $V = \{a\} \cup \{x_i, \neg x_i\}, \forall i = 1, 2, \ldots, n \cup \{C_{i1}, C_{i2}, C_{i3}\}, \forall i = 1, 2 \ldots m$, where $C_{ij}$ refers to the $j^{th}$ literal in the clause $C_i$.
   2. $E_1 = \{a, x_i\}, \forall i = 1, 2, \ldots n \cup \{a, \neg x_i\}, \forall i = 1, 2, \ldots n$.
   3. $E_2 = \{C_{i1}, C_{i2}\} \cup \{C_{i1}, C_{i3}\} \cup \{C_{i2}, C_{i3}\}, \forall i = 1, 2, \ldots, m$. 

5. **NP-completeness**

**Computational Complexity**
Theorem

**GRAPH 3-COLORABILITY** is NP-complete.

Proof

1. **GRAPH 3-COLORABILITY** is clearly in NP.
2. We reduce NAE3SAT to **GRAPH 3-COLORABILITY**.
3. Let \( \phi = C_1 \land C_2 \ldots C_m \) be a 3CNF formula over \( n \) variables and \( m \) clauses.
4. The corresponding instance of **GRAPH 3-COLORABILITY** is the graph \( G = \langle V, E \rangle \) constructed as follows:
   1. \( V = \{a\} \cup \{x_i, \neg x_i\}, \forall i = 1, 2, \ldots, n \cup \{C_{i1}, C_{i2}, C_{i3}\}, \forall i = 1, 2 \ldots m, \)
      where \( C_{ij} \) refers to the \( j^{th} \) literal in the clause \( C_i \).
   2. \( E_1 = \{a, x_i\}, \forall i = 1, 2, \ldots n \cup \{a, \neg x_i\}, \forall i = 1, 2, \ldots n. \)
   3. \( E_2 = \{C_{i1}, C_{i2}\} \cup \{C_{i1}, C_{i3}\} \cup \{C_{i2}, C_{i3}\}, \forall i = 1, 2, \ldots, m. \)
   4. \( E_3 = \cup \{C_{ij}, x_k\}, \)
Graph 3-Colorability is NP-complete.

Proof

1. Graph 3-Colorability is clearly in NP.
2. We reduce NAE3SAT to Graph 3-Colorability.
3. Let $\phi = C_1 \land C_2 \ldots C_m$ be a 3CNF formula over $n$ variables and $m$ clauses.
4. The corresponding instance of Graph 3-Colorability is the graph $G = \langle V, E \rangle$ constructed as follows:
   1. $V = \{a\} \cup \{x_i, \neg x_i\}, \forall i = 1, 2, \ldots, n \cup \{C_{i1}, C_{i2}, C_{i3}\}, \forall i = 1, 2 \ldots m$, where $C_{ij}$ refers to the $j$th literal in the clause $C_i$.
   2. $E_1 = \{a, x_i\}, \forall i = 1, 2, \ldots n \cup \{a, \neg x_i\}, \forall i = 1, 2, \ldots n$.
   3. $E_2 = \{C_{i1}, C_{i2}\} \cup \{C_{i1}, C_{i3}\} \cup \{C_{i2}, C_{i3}\}, \forall i = 1, 2, \ldots, m$.
   4. $E_3 = \cup \{C_{ij}, x_k\}, \forall j = 1, 2, 3,$
Theorem

**GRAPH 3-COLORABILITY** is **NP-complete**.

Proof

1. **GRAPH 3-COLORABILITY** is clearly in **NP**.
2. We reduce NAE3SAT to **GRAPH 3-COLORABILITY**.
3. Let $\phi = C_1 \land C_2 \ldots C_m$ be a 3CNF formula over $n$ variables and $m$ clauses.
4. The corresponding instance of **GRAPH 3-COLORABILITY** is the graph $G = \langle V, E \rangle$ constructed as follows:
   1. $V = \{a\} \cup \{x_i, \neg x_i\}, \forall i = 1, 2, \ldots, n \cup \{C_{i1}, C_{i2}, C_{i3}\}, \forall i = 1, 2 \ldots m,$
      where $C_{ij}$ refers to the $j^{th}$ literal in the clause $C_i$.
   2. $E_1 = \{a, x_i\}, \forall i = 1, 2, \ldots n \cup \{a, \neg x_i\}, \forall i = 1, 2, \ldots n.$
   3. $E_2 = \{C_{i1}, C_{i2}\} \cup \{C_{i1}, C_{i3}\} \cup \{C_{i2}, C_{i3}\}, \forall i = 1, 2, \ldots, m.$
   4. $E_3 = \cup\{C_{ij}, x_k\}, \forall j = 1, 2, 3, \forall i = 1, 2, \ldots, m, \forall k = 1, 2, \ldots n,$ if $C_{ij} = x_k.$
Theorem

**GRAPH 3-COLORABILITY** is **NP-complete**.

Proof

1. **GRAPH 3-COLORABILITY** is clearly in **NP**.
2. We reduce NAE3SAT to **GRAPH 3-COLORABILITY**.
3. Let \( \phi = C_1 \land C_2 \ldots C_m \) be a 3CNF formula over \( n \) variables and \( m \) clauses.
4. The corresponding instance of **GRAPH 3-COLORABILITY** is the graph \( G = \langle V, E \rangle \) constructed as follows:
   - \( V = \{a\} \cup \{x_i, \neg x_i\}, \forall i = 1, 2, \ldots, n \cup \{C_{i1}, C_{i2}, C_{i3}\}, \forall i = 1, 2 \ldots m \)
     where \( C_{ij} \) refers to the \( j \)th literal in the clause \( C_i \).
   - \( E_1 = \{a, x_i\}, \forall i = 1, 2, \ldots, n \cup \{a, \neg x_i\}, \forall i = 1, 2, \ldots n \).
   - \( E_2 = \{C_{i1}, C_{i2}\} \cup \{C_{i1}, C_{i3}\} \cup \{C_{i2}, C_{i3}\}, \forall i = 1, 2, \ldots, m \).
   - \( E_3 = \cup\{C_{ij}, x_k\}, \forall j = 1, 2, 3, \forall i = 1, 2, \ldots, m, \forall k = 1, 2, \ldots, n \), if \( C_{ij} = x_k \).
   - \( E_4 = \cup\{C_{ij}, \neg x_k\} \),
Graph 3-Colorability is NP-complete.

Proof

1. Graph 3-Colorability is clearly in NP.
2. We reduce NAE3SAT to Graph 3-Colorability.
3. Let $\phi = C_1 \wedge C_2 \ldots C_m$ be a 3CNF formula over $n$ variables and $m$ clauses.
4. The corresponding instance of Graph 3-Colorability is the graph $G = \langle V, E \rangle$ constructed as follows:
   1. $V = \{a\} \cup \{x_i, \neg x_i\}, \forall i = 1, 2, \ldots, n \cup \{C_{i1}, C_{i2}, C_{i3}\}, \forall i = 1, 2 \ldots m$, where $C_{ij}$ refers to the $j^{th}$ literal in the clause $C_i$.
   2. $E_1 = \{a, x_i\}, \forall i = 1, 2, \ldots n \cup \{a, \neg x_i\}, \forall i = 1, 2, \ldots n$.
   3. $E_2 = \{C_{i1}, C_{i2}\} \cup \{C_{i1}, C_{i3}\} \cup \{C_{i2}, C_{i3}\}, \forall i = 1, 2, \ldots, m$.
   4. $E_3 = \cup\{C_{ij}, x_k\}, \forall j = 1, 2, 3, \forall i = 1, 2, \ldots, m, \forall k = 1, 2, \ldots n$, if $C_{ij} = x_k$.
   5. $E_4 = \cup\{C_{ij}, \neg x_k\}, \forall j = 1, 2, 3$.
Theorem

**GRAPH 3-COLORABILITY is NP-complete.**

Proof

1. **GRAPH 3-COLORABILITY** is clearly in **NP**.
2. We reduce NAE3SAT to **GRAPH 3-COLORABILITY**.
3. Let \( \phi = C_1 \land C_2 \ldots C_m \) be a 3CNF formula over \( n \) variables and \( m \) clauses.
4. The corresponding instance of **GRAPH 3-COLORABILITY** is the graph \( G = \langle V, E \rangle \) constructed as follows:
   1. \( V = \{ a \} \cup \{ x_i, \neg x_i \}, \forall i = 1, 2, \ldots, n \cup \{ C_{i1}, C_{i2}, C_{i3} \}, \forall i = 1, 2 \ldots m, \) where \( C_{ij} \) refers to the \( j \)th literal in the clause \( C_i \).
   2. \( E_1 = \{ a, x_i \}, \forall i = 1, 2, \ldots n \cup \{ a, \neg x_i \}, \forall i = 1, 2, \ldots n. \)
   3. \( E_2 = \{ C_{i1}, C_{i2} \} \cup \{ C_{i1}, C_{i3} \} \cup \{ C_{i2}, C_{i3} \}, \forall i = 1, 2, \ldots, m. \)
   4. \( E_3 = \cup \{ C_{ij}, x_k \}, \forall j = 1, 2, 3, \forall i = 1, 2, \ldots, m, \forall k = 1, 2, \ldots n, \) if \( C_{ij} = x_k \).
   5. \( E_4 = \cup \{ C_{ij}, \neg x_k \}, \forall j = 1, 2, 3, \forall i = 1, 2, \ldots, m, \forall k = 1, 2, \ldots n, \) if \( C_{ij} = \neg x_k \).
Graph 3-Colorability is NP-complete.

Proof

1. **Graph 3-Colorability** is clearly in NP.
2. We reduce NAE3SAT to Graph 3-Colorability.
3. Let $\phi = C_1 \land C_2 \ldots C_m$ be a 3CNF formula over $n$ variables and $m$ clauses.
4. The corresponding instance of Graph 3-Colorability is the graph $G = \langle V, E \rangle$ constructed as follows:
   - $V = \{a\} \cup \{x_i, \neg x_i\}, \forall i = 1, 2, \ldots, n \cup \{C_{i1}, C_{i2}, C_{i3}\}, \forall i = 1, 2 \ldots m,$
     where $C_{ij}$ refers to the $j^{th}$ literal in the clause $C_i$.
   - $E_1 = \{a, x_i\}, \forall i = 1, 2, \ldots, n \cup \{a, \neg x_i\}, \forall i = 1, 2, \ldots, n.$
   - $E_2 = \{C_{i1}, C_{i2}\} \cup \{C_{i1}, C_{i3}\} \cup \{C_{i2}, C_{i3}\}, \forall i = 1, 2, \ldots, m.$
   - $E_3 = \{C_{ij}, x_k\}, \forall j = 1, 2, 3, \forall i = 1, 2, \ldots, m, \forall k = 1, 2, \ldots n,$ if $C_{ij} = x_k$.
   - $E_4 = \{C_{ij}, \neg x_k\}, \forall j = 1, 2, 3, \forall i = 1, 2, \ldots, m, \forall k = 1, 2, \ldots n,$ if $C_{ij} = \neg x_k$.
   - $E_5 = \{x_i, \neg x_i\}, \forall i = 1, 2, \ldots, n.$

NP-completeness

Computational Complexity
**Theorem**

**GRAPH 3-COLORABILITY is NP-complete.**

**Proof**

1. **GRAPH 3-COLORABILITY** is clearly in **NP**.
2. We reduce **NAE3SAT** to **GRAPH 3-COLORABILITY**.
3. Let \( \phi = C_1 \land C_2 \ldots C_m \) be a 3CNF formula over \( n \) variables and \( m \) clauses.
4. The corresponding instance of **GRAPH 3-COLORABILITY** is the graph \( G = \langle V, E \rangle \) constructed as follows:
   
   \( V = \{ a \} \cup \{ x_i, \neg x_i \}, \forall i = 1, 2, \ldots, n \cup \{ C_{i1}, C_{i2}, C_{i3} \}, \forall i = 1, 2 \ldots m, \) where \( C_{ij} \) refers to the \( j^{th} \) literal in the clause \( C_i \).

   1. \( E_1 = \{ a, x_i \}, \forall i = 1, 2, \ldots n \cup \{ a, \neg x_i \}, \forall i = 1, 2, \ldots n \).
   2. \( E_2 = \{ C_{i1}, C_{i2} \} \cup \{ C_{i1}, C_{i3} \} \cup \{ C_{i2}, C_{i3} \}, \forall i = 1, 2, \ldots, m. \)
   3. \( E_3 = \cup \{ C_{ij}, x_k \}, \forall j = 1, 2, 3, \forall i = 1, 2, \ldots, m, \forall k = 1, 2, \ldots n, \) if \( C_{ij} = x_k \).
   4. \( E_4 = \cup \{ C_{ij}, \neg x_k \}, \forall j = 1, 2, 3, \forall i = 1, 2, \ldots, m, \forall k = 1, 2, \ldots n, \) if \( C_{ij} = \neg x_k \).
   5. \( E_5 = \cup \{ x_i, \neg x_i \}, \forall i = 1, 2, \ldots n. \)
   6. \( E = E_1 \cup E_2 \cup E_3 \cup E_4 \cup E_5. \)
Example
Example

Construction for ... ∧ (x₁, ¬x₂, ¬x₃) ∧ ...
Completing the reduction

1 Assume that $G$ has a 3-coloring.
2 Without loss of generality, we can assume that $a$ has been colored 2. (Why?)
3 This means that for each pair $\{x_i, \neg x_i\}$, one of them has been assigned 0 and the other 1, i.e., we get a consistent assignment by setting literals assigned to 0 to false and literals assigned to 1 to true.
4 We will now argue that the assignment nae-satisfies every clause.
5 Can the assignment set every literal in a clause to true? How about false?
6 Now assume that $\phi$ has a nae-satisfying assignment.
7 Color the literals in $G$ as per this assignment and assign color 2 to vertex $a$.
8 Now focus on a clause triangle. The literal which is connected to a true literal is assigned the color 0 and the literal which is connected to a false literal is assigned the color 1. The remaining literal is assigned the color 2.
Completing the reduction
Completing the reduction

1. Assume that $G$ has a 3-coloring.
Argument

Completing the reduction

1. Assume that $G$ has a 3-coloring.
2. Without loss of generality, we can assume that $a$ has been colored 2. (Why?)
Completing the reduction

1. Assume that $G$ has a 3-coloring.
2. Without loss of generality, we can assume that $a$ has been colored 2. (Why?)
3. This means that for each pair $\{x_i, \neg x_i\}$, one of them has been assigned 0 and the other 1,
Completing the reduction

1. Assume that $G$ has a 3-coloring.
2. Without loss of generality, we can assume that $a$ has been colored 2. (Why?)
3. This means that for each pair $\{x_i, \neg x_i\}$, one of them has been assigned 0 and the other 1, i.e., we get a consistent assignment by setting literals assigned to 0 to false and literals assigned to 1 to true.
Completing the reduction

1. Assume that $G$ has a 3-coloring.
2. Without loss of generality, we can assume that $a$ has been colored 2. (Why?)
3. This means that for each pair $\{x_i, \neg x_i\}$, one of them has been assigned 0 and the other 1, i.e., we get a consistent assignment by setting literals assigned to 0 to \texttt{false} and literals assigned to 1 to \texttt{true}.
4. We will now argue that the assignment nae-satisfies every clause.
Completing the reduction

1. Assume that $G$ has a 3-coloring.
2. Without loss of generality, we can assume that $a$ has been colored 2. (Why?)
3. This means that for each pair $\{x_i, \neg x_i\}$, one of them has been assigned 0 and the other 1, i.e., we get a consistent assignment by setting literals assigned to 0 to $false$ and literals assigned to 1 to $true$.
4. We will now argue that the assignment nae-satisfies every clause.
5. Can the assignment set every literal in a clause to $true$?
Assume that $G$ has a 3-coloring.

Without loss of generality, we can assume that $a$ has been colored 2. (Why?)

This means that for each pair $\{x_i, \neg x_i\}$, one of them has been assigned 0 and the other 1, i.e., we get a consistent assignment by setting literals assigned to 0 to false and literals assigned to 1 to true.

We will now argue that the assignment nae-satisfies every clause.

Can the assignment set every literal in a clause to true? How about false?
Completing the reduction

1. Assume that $G$ has a 3-coloring.
2. Without loss of generality, we can assume that $a$ has been colored 2. (Why?)
3. This means that for each pair $\{x_i, \neg x_i\}$, one of them has been assigned 0 and the other 1, i.e., we get a consistent assignment by setting literals assigned to 0 to `false` and literals assigned to 1 to `true`.
4. We will now argue that the assignment nae-satisfies every clause.
5. Can the assignment set every literal in a clause to `true`? How about `false`?
6. Now assume that $\phi$ has a nae-satisfying assignment.
Completing the reduction

1. Assume that $G$ has a 3-coloring.
2. Without loss of generality, we can assume that $a$ has been colored 2. (Why?)
3. This means that for each pair $\{x_i, \neg x_i\}$, one of them has been assigned 0 and the other 1, i.e., we get a consistent assignment by setting literals assigned to 0 to $\text{false}$ and literals assigned to 1 to $\text{true}$.
4. We will now argue that the assignment nae-satisfies every clause.
5. Can the assignment set every literal in a clause to $\text{true}$? How about $\text{false}$?
6. Now assume that $\phi$ has a nae-satisfying assignment.
7. Color the literals in $G$ as per this assignment and assign color 2 to vertex $a$. 

NP-completeness
Computational Complexity
Completing the reduction

1. Assume that $G$ has a 3-coloring.
2. Without loss of generality, we can assume that $a$ has been colored 2. (Why?)
3. This means that for each pair $\{x_i, \neg x_i\}$, one of them has been assigned 0 and the other 1, i.e., we get a consistent assignment by setting literals assigned to 0 to \textit{false} and literals assigned to 1 to \textit{true}.
4. We will now argue that the assignment nae-satisfies every clause.
5. Can the assignment set every literal in a clause to \textit{true}? How about \textit{false}? 
6. Now assume that $\phi$ has a nae-satisfying assignment.
7. Color the literals in $G$ as per this assignment and assign color 2 to vertex $a$.
8. Now focus on a clause triangle.
Completing the reduction

1. Assume that $G$ has a 3-coloring.
2. Without loss of generality, we can assume that $a$ has been colored 2. (Why?)
3. This means that for each pair $\{x_i, \neg x_i\}$, one of them has been assigned 0 and the other 1, i.e., we get a consistent assignment by setting literals assigned to 0 to false and literals assigned to 1 to true.
4. We will now argue that the assignment nae-satisfies every clause.
5. Can the assignment set every literal in a clause to true? How about false?
6. Now assume that $\phi$ has a nae-satisfying assignment.
7. Color the literals in $G$ as per this assignment and assign color 2 to vertex $a$.
8. Now focus on a clause triangle. The literal which is connected to a true literal is assigned the color 0 and the literal which is connected to a false literal is assigned the color 1.
**Argument**

**Completing the reduction**

1. Assume that $G$ has a 3-coloring.
2. Without loss of generality, we can assume that $a$ has been colored 2. (Why?)
3. This means that for each pair $\{x_i, \neg x_i\}$, one of them has been assigned 0 and the other 1, i.e., we get a consistent assignment by setting literals assigned to 0 to **false** and literals assigned to 1 to **true**.
4. We will now argue that the assignment nae-satisfies every clause.
5. Can the assignment set every literal in a clause to **true**? How about **false**?
6. Now assume that $\phi$ has a nae-satisfying assignment.
7. Color the literals in $G$ as per this assignment and assign color 2 to vertex $a$.
8. Now focus on a clause triangle.
The literal which is connected to a **true** literal is assigned the color 0 and the literal which is connected to a **false** literal is assigned the color 1. The remaining literal is assigned the color 2.
MaxCut

Definition

A cut in an undirected graph $G = (V, E)$ is a partition of vertices into two non-empty sets $S$ and $V - S$.

The size of a cut $(S, V - S)$ is the number of edges between $S$ and $V - S$.

Definition

Input: An undirected graph $G = \langle V, E \rangle$ and a number $K$.

Query: Is there a cut of size at least $K$ in $G$?
MaxCut

Definition

A cut in an undirected graph \( G = (V, E) \) is a partition of vertices into two non-empty sets \( S \) and \( V - S \). The size of a cut \((S, V - S)\) is the number of edges between \( S \) and \( V - S \).

Definition

Input: An undirected graph \( G = \langle V, E \rangle \) and a number \( K \).

Query: Is there a cut of size at least \( K \) in \( G \)?
A cut in an undirected graph $G = (V, E)$ is a partition of vertices into two non-empty sets $S$ and $V - S$. 
A cut in an undirected graph $G = (V, E)$ is a partition of vertices into two non-empty sets $S$ and $V - S$.

The size of a cut $(S, V - S)$ is the number of edges between $S$ and $V - S$. 
**MaxCut**

**Definition**

A cut in an undirected graph $G = (V, E)$ is a partition of vertices into two non-empty sets $S$ and $V - S$.

The size of a cut $(S, V - S)$ is the number of edges between $S$ and $V - S$. 
MaxCut

Definition

A cut in an undirected graph $G = (V, E)$ is a partition of vertices into two non-empty sets $S$ and $V - S$.

The size of a cut $(S, V - S)$ is the number of edges between $S$ and $V - S$.

Definition

Input: An undirected graph $G = \langle V, E \rangle$ and a number $K$. 

NP-completeness
MaxCut

Definition
A cut in an undirected graph $G = (V, E)$ is a partition of vertices into two non-empty sets $S$ and $V - S$.

The size of a cut $(S, V - S)$ is the number of edges between $S$ and $V - S$.

**Definition**

**Input:** An undirected graph $G = \langle V, E \rangle$ and a number $K$.

**Query:** Is there a cut of size at least $K$ in $G$?
Theorem

\[ \text{MAX CUT is NP-complete}. \]

Proof

1. MAX CUT is clearly in \( \text{NP} \).
2. We reduce NAE3SAT to \( \text{MAX CUT} \).
3. Let \( \phi = C_1 \land C_2 \ldots C_m \) denote a 3CNF formula over \( n \) variables and \( m \) clauses.
4. We construct the graph \( G = \langle V, E \rangle \) as follows:
   1. \( V = \{ x_1, x_2, \ldots, x_n \} \cup \{ \neg x_1, \neg x_2, \ldots, \neg x_n \} \).
   2. \( E_1 = \) triangles from the three literals in each clause (parallel edges if needed).
   3. \( E_2 = n_{i} \) edges from \( x_i \) to \( \neg x_i \), where \( n_{i} \) is the number of occurrences of \( x_i \) and \( \neg x_i \) across all the clauses.
   4. \( E = E_1 \cup E_2 \).
5. Set \( K = 5 \cdot m \).
NP-completeness

Theorem

**Theorem**

1. **M**ax **C**ut is clearly in **NP**.

2. We reduce **NAE3SAT** to **M**ax **C**ut.

3. Let $\phi = C_1 \land C_2 \ldots C_m$ denote a 3CNF formula over $n$ variables and $m$ clauses.

4. We construct the graph $G = \langle V, E \rangle$ as follows:
   1. $V = \{x_1, x_2, \ldots, x_n\} \cup \{\neg x_1, \neg x_2, \ldots, \neg x_n\}$.
   2. $E_1 =$ triangles from the three literals in each clause (parallel edges if needed).
   3. $E_2 =$ $n_i$ edges from $x_i$ to $\neg x_i$, where $n_i$ is the number of occurrences of $x_i$ and $\neg x_i$ across all the clauses.
   4. $E = E_1 \cup E_2$.
   5. Set $K = 5 \cdot m$.
Theorem

MaxCut is NP-complete.
Theorem

MaxCut is NP-complete.

Proof

1. MaxCut is clearly in NP.
2. We reduce NAE3SAT to MaxCut.
3. Let \( \phi = C_1 \land C_2 \ldots C_m \) denote a 3CNF formula over \( n \) variables and \( m \) clauses.
4. We construct the graph \( G = \langle V, E \rangle \) as follows:
   1. \( V = \{ x_1, x_2, \ldots, x_n \} \cup \{ \neg x_1, \neg x_2, \ldots, \neg x_n \} \).
   2. \( E_1 = \) triangles from the three literals in each clause (parallel edges if needed).
   3. \( E_2 = n_i \) edges from \( x_i \) to \( \neg x_i \), where \( n_i \) is the number of occurrences of \( x_i \) and \( \neg x_i \) across all the clauses.
   4. \( E = E_1 \cup E_2 \).
   5. Set \( K = 5 \cdot m \).
Theorem

\textbf{MaxCut is NP-complete.}

Proof

1 \textbf{MaxCut is clearly in NP.}
**Theorem**

\textbf{MaxCut} is \textbf{NP-complete}.

**Proof**

1. \textbf{MaxCut} is clearly in \textbf{NP}.
2. We reduce NAE3SAT to \textbf{MaxCut}.
Theorem

\textbf{MaxCut is NP-complete.}

Proof

1. \textbf{MaxCut} is clearly in \textbf{NP}.
2. We reduce \textbf{NAE3SAT} to \textbf{MaxCut}.
3. Let $\phi = C_1 \land C_2 \ldots C_m$ denote a 3CNF formula over $n$ variables and $m$ clauses.
Theorem

MaxCut is NP-complete.

Proof

1. MaxCut is clearly in NP.
2. We reduce NAE3SAT to MaxCut.
3. Let $\phi = C_1 \land C_2 \ldots C_m$ denote a 3CNF formula over $n$ variables and $m$ clauses.
4. We construct the graph $G = \langle V, E \rangle$ as follows:


**Theorem**

MaxCut is NP-complete.

**Proof**

1. MaxCut is clearly in NP.
2. We reduce NAE3SAT to MaxCut.
3. Let $\phi = C_1 \land C_2 \ldots C_m$ denote a 3CNF formula over $n$ variables and $m$ clauses.
4. We construct the graph $G = \langle V, E \rangle$ as follows:
   1. $V = \{x_1, x_2, \ldots, x_n\}$
**Theorem**

\( \text{MaxCut} \) is **NP-complete**.

**Proof**

1. \( \text{MaxCut} \) is clearly in **NP**.
2. We reduce NAE3SAT to \( \text{MaxCut} \).
3. Let \( \phi = C_1 \land C_2 \ldots C_m \) denote a 3CNF formula over \( n \) variables and \( m \) clauses.
4. We construct the graph \( G = \langle V, E \rangle \) as follows:
   1. \( V = \{x_1, x_2, \ldots x_n\} \cup \{-x_1, -x_2, \ldots -x_n\} \).
Complexity

Theorem

MaxCut is NP-complete.

Proof

1. MaxCut is clearly in NP.
2. We reduce NAE3SAT to MaxCut.
3. Let \( \phi = C_1 \land C_2 \ldots C_m \) denote a 3CNF formula over \( n \) variables and \( m \) clauses.
4. We construct the graph \( G = \langle V, E \rangle \) as follows:
   1. \( V = \{x_1, x_2, \ldots x_n\} \cup \{\neg x_1, \neg x_2, \ldots \neg x_n\} \).
   2. \( E_1 = \) triangles from the three literals in each clause
Theorem

MaxCut is \textbf{NP-complete}.

Proof

1. MaxCut is clearly in \textbf{NP}.
2. We reduce NAE3SAT to MaxCut.
3. Let $\phi = C_1 \land C_2 \ldots C_m$ denote a 3CNF formula over $n$ variables and $m$ clauses.
4. We construct the graph $G = \langle V, E \rangle$ as follows:
   1. $V = \{x_1, x_2, \ldots x_n\} \cup \{\neg x_1, \neg x_2, \ldots \neg x_n\}$.
   2. $E_1 = \text{triangles from the three literals in each clause (parallel edges if needed)}.$
**Theorem**

\textbf{MaxCut is NP-complete.}

**Proof**

1. \textbf{MaxCut} is clearly in \textbf{NP}.
2. We reduce NAE3SAT to \textbf{MaxCut}.
3. Let \( \phi = C_1 \land C_2 \ldots C_m \) denote a 3CNF formula over \( n \) variables and \( m \) clauses.
4. We construct the graph \( G = \langle V, E \rangle \) as follows:
   1. \( V = \{x_1, x_2, \ldots x_n\} \cup \{\neg x_1, \neg x_2, \ldots \neg x_n\} \).
   2. \( E_1 = \) triangles from the three literals in each clause (parallel edges if needed).
   3. \( E_2 = n_i \) edges from \( x_i \) to \( \neg x_i \), where \( n_i \) is the number of occurrences of \( x_i \) and \( \neg x_i \) across all the clauses.
Theorem

*MaxCut* is NP-complete.

Proof

1. **MaxCut** is clearly in NP.
2. We reduce NAE3SAT to MaxCut.
3. Let $\phi = C_1 \wedge C_2 \ldots C_m$ denote a 3CNF formula over $n$ variables and $m$ clauses.
4. We construct the graph $G = \langle V, E \rangle$ as follows:
   1. $V = \{x_1, x_2, \ldots x_n\} \cup \{\neg x_1, \neg x_2, \ldots \neg x_n\}$.
   2. $E_1 =$ triangles from the three literals in each clause (parallel edges if needed).
   3. $E_2 = n_i$ edges from $x_i$ to $\neg x_i$, where $n_i$ is the number of occurrences of $x_i$ and $\neg x_i$ across all the clauses.
   4. $E = E_1 \cup E_2$. 
Theorem

**MaxCut is NP-complete.**

Proof

1. **MaxCut** is clearly in **NP**.
2. We reduce NAE3SAT to **MaxCut**.
3. Let \( \phi = C_1 \land C_2 \ldots C_m \) denote a 3CNF formula over \( n \) variables and \( m \) clauses.
4. We construct the graph \( G = \langle V, E \rangle \) as follows:
   1. \( V = \{x_1, x_2, \ldots x_n\} \cup \{-x_1, -x_2, \ldots -x_n\} \).
   2. \( E_1 = \) triangles from the three literals in each clause (parallel edges if needed).
   3. \( E_2 = n_i \) edges from \( x_i \) to \(-x_i\), where \( n_i \) is the number of occurrences of \( x_i \) and \(-x_i\) across all the clauses.
   4. \( E = E_1 \cup E_2 \).
   5. Set \( K = 5 \cdot m \).
Example

Let $\phi = (x_1 \lor x_2) \land (x_1 \lor \neg x_3) \land (\neg x_1 \lor \neg x_2 \lor x_3)$
Example

Let $\phi =$

$\phi = (x_1 \lor x_2) \land (x_1 \lor \neg x_3) \land (\neg x_1 \lor \neg x_2 \lor x_3)$
Example

Let $\phi = (x_1 \lor x_2) \land (x_1 \lor \neg x_3) \land (\neg x_1 \lor \neg x_2 \lor x_3)$
Example

Let $\phi = (x_1 \lor x_2) \land (x_1 \lor \neg x_3) \land (\neg x_1 \lor \neg x_2 \lor x_3) \equiv$ $(x_1 \lor x_2 \lor x_2) \land (x_1 \lor \neg x_3 \lor \neg x_3) \land (\neg x_1 \lor \neg x_2 \lor x_3)$
Example

Let \( \phi = (x_1 \lor x_2) \land (x_1 \lor \neg x_3) \land (\neg x_1 \lor \neg x_2 \lor x_3) \equiv (x_1 \lor x_2 \lor x_2) \land (x_1 \lor \neg x_3 \lor \neg x_3) \land (\neg x_1 \lor \neg x_2 \lor x_3) \)
Argument - Part I

Lemma

Assume that $G$ has a cut of at least $5 \cdot m$. Then $\phi$ has a nae-satisfying assignment.

Proof

1. We can safely assume that $x_i$ and $\neg x_i$ are on opposite sides of the cut. Why?
2. The edges between the $x_i$ and $\neg x_i$ contribute exactly $3 \cdot m$ edges to the cut. Why?
3. The remaining $2 \cdot m$ or more edges must come from the clause triangles.
4. Each clause triangle can contribute at most 2 edges. Why?
5. It follows that every clause triangle is cut and that the total number of cut edges is exactly $5 \cdot m$.
6. Arbitrarily assign true to the literals on one side of the cut and false to the rest.
7. Clearly, this is a consistent assignment.
8. Since each triangle is cut, it means that each clause has at least one literal set to true and at least one set to false, i.e., the assignment is nae-satisfying.
Lemma

Assume that $G$ has a cut of at least $5 \cdot m$. Then $\phi$ has a nae-satisfying assignment.

Proof

1. We can safely assume that $x_i$ and $\neg x_i$ are on opposite sides of the cut. Why?
2. The edges between the $x_i$ and $\neg x_i$ contribute exactly $3 \cdot m$ edges to the cut. Why?
3. The remaining $2 \cdot m$ or more edges must come from the clause triangles.
4. Each clause triangle can contribute at most 2 edges. Why?
5. It follows that every clause triangle is cut and that the total number of cut edges is exactly $5 \cdot m$.
6. Arbitrarily assign true to the literals on one side of the cut and false to the rest.
7. Clearly, this is a consistent assignment.
8. Since each triangle is cut, it means that each clause has at least one literal set to true and at least one set to false, i.e., the assignment is nae-satisfying.
Lemma

Assume that $G$ has a cut of at least $5 \cdot m$. 

Proof

1. We can safely assume that $x_i$ and $\neg x_i$ are on opposite sides of the cut. Why?
2. The edges between the $x_i$ and $\neg x_i$ contribute exactly $3 \cdot m$ edges to the cut. Why?
3. The remaining $2 \cdot m$ or more edges must come from the clause triangles.
4. Each clause triangle can contribute at most 2 edges. Why?
5. It follows that every clause triangle is cut and that the total number of cut edges is exactly $5 \cdot m$.
6. Arbitrarily assign true to the literals on one side of the cut and false to the rest.
7. Clearly, this is a consistent assignment.
8. Since each triangle is cut, it means that each clause has at least one literal set to true and at least one set to false, i.e., the assignment is nae-satisfying.
Lemma

Assume that $G$ has a cut of at least $5 \cdot m$. Then $\phi$ has a nae-satisfying assignment.
Lemma

*Assume that $G$ has a cut of at least $5 \cdot m$. Then $\phi$ has a nae-satisfying assignment.*

Proof
Lemma

Assume that $G$ has a cut of at least $5 \cdot m$. Then $\phi$ has a nae-satisfying assignment.

Proof

1. We can safely assume that $x_i$ and $\neg x_i$ are on opposite sides of the cut. Why?
Lemma

Assume that $G$ has a cut of at least $5 \cdot m$. Then $\phi$ has a nae-satisfying assignment.

Proof

1. We can safely assume that $x_i$ and $\neg x_i$ are on opposite sides of the cut. Why?
2. The edges between the $x_i$ and $\neg x_i$ contribute exactly $3 \cdot m$ edges to the cut. Why?
Lemma

Assume that $G$ has a cut of at least $5 \cdot m$. Then $\phi$ has a nae-satisfying assignment.

Proof

1. We can safely assume that $x_i$ and $\neg x_i$ are on opposite sides of the cut. Why?
2. The edges between the $x_i$ and $\neg x_i$ contribute exactly $3 \cdot m$ edges to the cut. Why?
3. The remaining $2 \cdot m$ or more edges must come from the clause triangles.
Lemma

Assume that $G$ has a cut of at least $5 \cdot m$. Then $\phi$ has a nae-satisfying assignment.

Proof

1. We can safely assume that $x_i$ and $\neg x_i$ are on opposite sides of the cut. Why?
2. The edges between the $x_i$ and $\neg x_i$ contribute exactly $3 \cdot m$ edges to the cut. Why?
3. The remaining $2 \cdot m$ or more edges must come from the clause triangles.
4. Each clause triangle can contribute at most 2 edges. Why?
Lemma

Assume that $G$ has a cut of at least $5 \cdot m$. Then $\varphi$ has a nae-satisfying assignment.

Proof

1. We can safely assume that $x_i$ and $\neg x_i$ are on opposite sides of the cut. Why?
2. The edges between the $x_i$ and $\neg x_i$ contribute exactly $3 \cdot m$ edges to the cut. Why?
3. The remaining $2 \cdot m$ or more edges must come from the clause triangles.
4. Each clause triangle can contribute at most 2 edges. Why?
5. It follows that every clause triangle is cut and that the total number of cut edges is exactly $5 \cdot m$. 
Argument - Part I

Lemma

Assume that $G$ has a cut of at least $5 \cdot m$. Then $\phi$ has a nae-satisfying assignment.

Proof

1. We can safely assume that $x_i$ and $\neg x_i$ are on opposite sides of the cut. Why?
2. The edges between the $x_i$ and $\neg x_i$ contribute exactly $3 \cdot m$ edges to the cut. Why?
3. The remaining $2 \cdot m$ or more edges must come from the clause triangles.
4. Each clause triangle can contribute at most 2 edges. Why?
5. It follows that every clause triangle is cut and that the total number of cut edges is exactly $5 \cdot m$.
6. Arbitrarily assign true to the literals on one side of the cut and false to the rest.
**Lemma**

Assume that $G$ has a cut of at least $5 \cdot m$. Then $\phi$ has a nae-satisfying assignment.

**Proof**

1. We can safely assume that $x_i$ and $\neg x_i$ are on opposite sides of the cut. Why?
2. The edges between the $x_i$ and $\neg x_i$ contribute exactly $3 \cdot m$ edges to the cut. Why?
3. The remaining $2 \cdot m$ or more edges must come from the clause triangles.
4. Each clause triangle can contribute at most 2 edges. Why?
5. It follows that every clause triangle is cut and that the total number of cut edges is exactly $5 \cdot m$.
6. Arbitrarily assign true to the literals on one side of the cut and false to the rest.
7. Clearly, this is a consistent assignment.
Lemma

Assume that $G$ has a cut of at least $5 \cdot m$. Then $\phi$ has a nae-satisfying assignment.

Proof

1. We can safely assume that $x_i$ and $\neg x_i$ are on opposite sides of the cut. Why?
2. The edges between the $x_i$ and $\neg x_i$ contribute exactly $3 \cdot m$ edges to the cut. Why?
3. The remaining $2 \cdot m$ or more edges must come from the clause triangles.
4. Each clause triangle can contribute at most 2 edges. Why?
5. It follows that every clause triangle is cut and that the total number of cut edges is exactly $5 \cdot m$.
6. Arbitrarily assign true to the literals on one side of the cut and false to the rest.
7. Clearly, this is a consistent assignment.
8. Since each triangle is cut, it means that each clause has at least one literal set to true and at least one set to false, i.e., the assignment is nae-satisfying.
Argument - Part II

Lemma

Assume that $\phi$ has a nae-satisfying assignment.

Then $G$ has a cut of at least $5 \cdot m$.

Proof

1. Let $S$ denote the set of vertices corresponding to literals that are assigned true.
2. We will argue that the cut $(S, V - S)$ has at least $5 \cdot m$ edges.
3. Since the assignment is consistent, $x_i$ and $\neg x_i$ are on opposite sides of the cut, i.e., these vertices contribute $3 \cdot m$ edges to the cut.
4. Since the assignment is nae-satisfying, every triangle is cut and thus an additional $2 \cdot m$ edges are contributed to the cut.
5. It follows that the cut $(S, V - S)$ has at least $5 \cdot m$ edges; in fact, it has exactly $5 \cdot m$ edges.

NP-completeness

Computational Complexity
Lemma

Assume that $\phi$ has a nae-satisfying assignment. Then $G$ has a cut of at least $5 \cdot m$.

Proof
1. Let $S$ denote the set of vertices corresponding to literals that are assigned true.
2. We will argue that the cut $(S, V - S)$ has at least $5 \cdot m$ edges.
3. Since the assignment is consistent, $x_i$ and $\neg x_i$ are on opposite sides of the cut, i.e., these vertices contribute $3 \cdot m$ edges to the cut.
4. Since the assignment is nae-satisfying, every triangle is cut and thus an additional $2 \cdot m$ edges are contributed to the cut.
5. It follows that the cut $(S, V - S)$ has at least $5 \cdot m$ edges; in fact, it has exactly $5 \cdot m$ edges.
Lemma

Assume that $\phi$ has a nae-satisfying assignment.
Lemma

Assume that $\phi$ has a nae-satisfying assignment. Then $G$ has a cut of at least $5 \cdot m$. 
Lemma

Assume that $\phi$ has a nae-satisfying assignment. Then $G$ has a cut of at least $5 \cdot m$.

Proof
Lemma

Assume that $\phi$ has a nae-satisfying assignment. Then $G$ has a cut of at least $5 \cdot m$.

Proof

1. Let $S$ denote the set of vertices corresponding to literals that are assigned true.
Lemma

Assume that $\phi$ has a nae-satisfying assignment. Then $G$ has a cut of at least $5 \cdot m$.

Proof

1. Let $S$ denote the set of vertices corresponding to literals that are assigned true.
2. We will argue that the cut $(S, V - S)$ has at least $5 \cdot m$ edges.
Lemma

Assume that $\phi$ has a nae-satisfying assignment. Then $G$ has a cut of at least $5 \cdot m$.

Proof

1. Let $S$ denote the set of vertices corresponding to literals that are assigned true.
2. We will argue that the cut $(S, V - S)$ has at least $5 \cdot m$ edges.
3. Since the assignment is consistent, $x_i$ and $\neg x_i$ are on opposite sides of the cut,
Lemma

Assume that $\phi$ has a nae-satisfying assignment. Then $G$ has a cut of at least $5 \cdot m$.

Proof

1. Let $S$ denote the set of vertices corresponding to literals that are assigned $\text{true}$.
2. We will argue that the cut $(S, V - S)$ has at least $5 \cdot m$ edges.
3. Since the assignment is consistent, $x_i$ and $\neg x_i$ are on opposite sides of the cut, i.e., these vertices contribute $3 \cdot m$ edges to the cut.
Lemma

Assume that $\phi$ has a nae-satisfying assignment. Then $G$ has a cut of at least $5 \cdot m$.

Proof

1. Let $S$ denote the set of vertices corresponding to literals that are assigned true.
2. We will argue that the cut $(S, V - S)$ has at least $5 \cdot m$ edges.
3. Since the assignment is consistent, $x_i$ and $\neg x_i$ are on opposite sides of the cut, i.e., these vertices contribute $3 \cdot m$ edges to the cut.
4. Since the assignment is nae-satisfying, every triangle is cut and thus an additional $2 \cdot m$ edges are contributed to the cut.
Lemma

Assume that $\phi$ has a nae-satisfying assignment. Then $G$ has a cut of at least $5 \cdot m$.

Proof

1. Let $S$ denote the set of vertices corresponding to literals that are assigned true.
2. We will argue that the cut $(S, V - S)$ has at least $5 \cdot m$ edges.
3. Since the assignment is consistent, $x_i$ and $\neg x_i$ are on opposite sides of the cut, i.e., these vertices contribute $3 \cdot m$ edges to the cut.
4. Since the assignment is nae-satisfying, every triangle is cut and thus an additional $2 \cdot m$ edges are contributed to the cut.
5. It follows that the cut $(S, V - S)$ has at least $5 \cdot m$ edges;
**Lemma**

Assume that $\phi$ has a nae-satisfying assignment. Then $G$ has a cut of at least $5 \cdot m$.

**Proof**

1. Let $S$ denote the set of vertices corresponding to literals that are assigned true.
2. We will argue that the cut $(S, V - S)$ has at least $5 \cdot m$ edges.
3. Since the assignment is consistent, $x_i$ and $\neg x_i$ are on opposite sides of the cut, i.e., these vertices contribute $3 \cdot m$ edges to the cut.
4. Since the assignment is nae-satisfying, every triangle is cut and thus an additional $2 \cdot m$ edges are contributed to the cut.
5. It follows that the cut $(S, V - S)$ has at least $5 \cdot m$ edges; in fact, it has exactly $5 \cdot m$ edges.
Max-Bisection

Definition

Input: An undirected graph \( G = \langle V, E \rangle \) and a number \( K \).

Query: Is there a cut \((S, V - S)\) of size at least \( K \) in \( G \), such that \( |S| = |V - S| \)?

Example: NP-completeness
Max-Bisection

Definition

Input: An undirected graph $G = \langle V, E \rangle$ and a number $K$.

Query: Is there a cut $(S, V - S)$ of size at least $K$ in $G$, such that $|S| = |V - S|$?
Max-Bisection

**Definition**

**Input:** An undirected graph $G = \langle V, E \rangle$ and a number $K$. 
Max-Bisection

**Definition**

**Input**: An undirected graph \( G = (V, E) \) and a number \( K \).

**Query**: Is there a cut \((S, V - S)\) of size at least \(K\) in \(G\), such that \(|S| = |V - S|\)?
**Max-Bisection**

**Definition**

**Input:** An undirected graph $G = \langle V, E \rangle$ and a number $K$.

**Query:** Is there a cut $(S, V - S)$ of size at least $K$ in $G$, such that $|S| = |V - S|$?

**Example**

[Blank space for example]
**Max-Bisection**

**Definition**

**Input:** An undirected graph $G = \langle V, E \rangle$ and a number $K$.

**Query:** Is there a cut $(S, V - S)$ of size at least $K$ in $G$, such that $|S| = |V - S|$?

**Example**

![Graph Example](image)
Theorem MX-BISECTION is NP-complete.

Proof
1. MX-BISECTION is clearly in NP.
2. We reduce MX-CUT to MX-BISECTION.
3. Given an instance \((G = \langle V, E \rangle, K)\) of MX-CUT, construct an instance of MX-BISECTION \((G' = \langle V', E' \rangle, K')\) as follows:
   1. \(V' = V \cup \{r_1, r_2, \ldots, r_{|V|}\}\).
   2. \(E' = E\).
   3. \(K' = K\).
Theorem

$\text{MAX-}B\text{-SECTION}$ is NP-complete.

Proof

1. $\text{MAX-}B\text{-SECTION}$ is clearly in $\text{NP}$.

2. We reduce $\text{MAX-CUT}$ to $\text{MAX-}B\text{-SECTION}$.

3. Given an instance $(G = \langle V, E \rangle, K)$ of $\text{MAX-CUT}$, construct an instance of $\text{MAX-}B\text{-SECTION}$ $(G' = \langle V', E' \rangle, K')$ as follows:

   1. $V' = V \cup \{r_1, r_2, \ldots, r_{|V|}\}$

   2. $E' = E$

   3. $K' = K$
Theorem

**Max-Bisection is NP-complete.**
Theorem

**Max-Bisection is NP-complete.**

Proof
Theorem

**MAX-BISECTION** is **NP-complete**.

Proof

1. **MAX-BISECTION** is clearly in **NP**.
Theorem

Max-Bisection is NP-complete.

Proof

1. Max-Bisection is clearly in NP.
2. We reduce MaxCut to Max-Bisection.
Theorem

MAX-BISECTION is NP-complete.

Proof

1. MAX-BISECTION is clearly in NP.
2. We reduce MAXCUT to MAX-BISECTION.
3. Given an instance \((G = \langle V, E \rangle, K)\) of MAXCUT, construct an instance of MAX-BISECTION \((G' = \langle V', E' \rangle, K')\) as follows:
Theorem

**MAX-BISECTION** is NP-complete.

Proof

1. **MAX-BISECTION** is clearly in NP.
2. We reduce **MAX-CUT** to **MAX-BISECTION**.
3. Given an instance \((G = \langle V, E \rangle, K)\) of **MAX-CUT**, construct an instance of **MAX-BISECTION** \((G' = \langle V', E' \rangle, K')\) as follows:
   1. \(V' = V \cup \{r_1, r_2, \ldots, r_{|V|}\}\).
Theorem

\textbf{Max-Bisection is NP-complete.}

Proof

1. \textbf{Max-Bisection} is clearly in \textbf{NP}.
2. We reduce \textbf{MaxCut} to \textbf{Max-Bisection}.
3. Given an instance \((G = \langle V, E \rangle, K)\) of \textbf{MaxCut}, construct an instance of \textbf{Max-Bisection} \((G' = \langle V', E' \rangle, K')\) as follows:
   1. \(V' = V \cup \{r_1, r_2, \ldots, r_{|V|}\}\).
   2. \(E' = E\).
Theorem

**Max-Bisection is NP-complete.**

Proof

1. **Max-Bisection** is clearly in **NP**.
2. We reduce **MaxCut** to **Max-Bisection**.
3. Given an instance \((G = \langle V, E \rangle, K)\) of **MaxCut**, construct an instance of **Max-Bisection** \((G' = \langle V', E' \rangle, K')\) as follows:
   1. \(V' = V \cup \{r_1, r_2, \ldots, r_{|V|}\}\).
   2. \(E' = E\).
   3. \(K' = K\).
Completing the argument it is not hard to see that every cut in $G$ can be made into a bisection in $G'$ by appropriately distributing the isolated vertices.
Completing the argument

It is not hard to see that every cut in $G$ can be made into a bisection in $G'$ by appropriately distributing the isolated vertices.
Completing the argument

It is not hard to see that every cut in $G$ can be made into a bisection in $G'$ by appropriately distributing the isolated vertices.
Definition
Input: An undirected graph $G = \langle V, E \rangle$ and a number $K$.
Query: Is there a cut $(S, V - S)$ of size at most $K$ in $G$, such that $|S| = |V - S|$?

Bisection-Width imposes an additional constraint on $\text{MINCUT}$, just as $\text{MAXBISECTION}$ imposes an additional constraint on $\text{MxFCUT}$.

Example
NP-completeness
Computational Complexity
Bisection-Width

Definition

Input: An undirected graph $G = (V, E)$ and a number $K$.

Query: Is there a cut $(S, V - S)$ of size at most $K$ in $G$, such that $|S| = |V - S|$?

Bisection-Width imposes an additional constraint on $\text{MINCUT}$, just as $\text{MAXBISECTION}$ imposes an additional constraint on $\text{MxCUT}$.

Example

NP-completeness

Computational Complexity
Bisection-Width

**Definition**

**Input:** An undirected graph \( G = \langle V, E \rangle \) and a number \( K \).
Bisection-Width

**Definition**

**Input:** An undirected graph $G = \langle V, E \rangle$ and a number $K$.

**Query:** Is there a cut $(S, V - S)$ of size at most $K$ in $G$, such that $|S| = |V - S|$?
Definition

**Input:** An undirected graph $G = \langle V, E \rangle$ and a number $K$.

**Query:** Is there a cut $(S, V - S)$ of size at most $K$ in $G$, such that $|S| = |V - S|$?

**Bisection-Width** imposes an additional constraint on **MINCUT**, just as **MAX-Bisection** imposes an additional constraint on **MAXCUT**.
Bisection-Width

**Definition**

**Input:** An undirected graph $G = \langle V, E \rangle$ and a number $K$.

**Query:** Is there a cut $(S, V - S)$ of size at most $K$ in $G$, such that $|S| = |V - S|$?

**Bisection-Width** imposes an additional constraint on **MINCUT**, just as **MAX-Bisection** imposes an additional constraint on **MAXCUT**.
Bisection-Width

**Definition**

**Input:** An undirected graph $G = \langle V, E \rangle$ and a number $K$.

**Query:** Is there a cut $(S, V - S)$ of size at most $K$ in $G$, such that $|S| = |V - S|$?

**Bisection-Width** imposes an additional constraint on **MINCUT**, just as **MAX-Bisection** imposes an additional constraint on **MAXCUT**.

**Example**

![Graph Example](image)
Theorem

**Theorem**

- *Bisection*-width is NP-complete.

**Proof**

1. **Bisection**-width is clearly in **NP**.
2. We reduce **Max Bisection** to **Bisection**-width.
3. Let \((G = (V,E), K)\) denote an instance of **Max Bisection**.
4. Without loss of generality, assume that \(|V| = 2n\).
5. The corresponding instance of **Bisection**-width is: \((G_c = (V,E_c), n^2 - K)\).
6. It is not hard to see that \(G\) has a bisection of size \(K\) or more if and only if \(G_c\) has a bisection of size \(n^2 - K\) or less.
Theorem

1. Bisection- WIDTH is clearly in NP.

2. We reduce MAX-BISECTION to BISECTION- WIDTH.

3. Let $(G = \langle V, E \rangle, K)$ denote an instance of MAX-BISECTION.

4. Without loss of generality, assume that $|V| = 2 \cdot n$.

5. The corresponding instance of BISECTION- WIDTH is: $(G_c = \langle V, E_c \rangle, n^2 - K)$.

6. It is not hard to see that $G$ has a bisection of size $K$ or more if and only if $G_c$ has a bisection of size $n^2 - K$ or less.

NP-completeness - Computational Complexity
Theorem

\textbf{Bisection-Width is NP-complete.}
Theorem

BISECTION-WIDTH is NP-complete.

Proof
Theorem

**BISECTION-WIDTH is NP-complete.**

Proof

1. **BISECTION-WIDTH is clearly in NP.**
Complexity

Theorem

\textsc{Bisection-Width} is \textbf{NP-complete}.

Proof

1. \textsc{Bisection-Width} is clearly in \textbf{NP}.
2. We reduce \textsc{Max-Bisection} to \textsc{Bisection-Width}.
**Theorem**

**Bisection-Width is NP-complete.**

**Proof**

1. **Bisection-Width** is clearly in **NP**.
2. We reduce **Max-Bisection** to **Bisection-Width**.
3. Let \((G = \langle V, E \rangle, K)\) denote an instance of **Max-Bisection**.
Theorem

**Bisection-Width is NP-complete.**

Proof

1. **Bisection-Width** is clearly in **NP**.
2. We reduce **Max-Bisection** to **Bisection-Width**.
3. Let \( (G = \langle V, E \rangle, K) \) denote an instance of **Max-Bisection**.
4. Without loss of generality, assume that \( |V| = 2 \cdot n \).
Theorem

**Bisection-Width is NP-complete.**

Proof

1. **Bisection-Width** is clearly in **NP**.
2. We reduce **Max-Bisection** to **Bisection-Width**.
3. Let \((G = \langle V, E \rangle, K)\) denote an instance of **Max-Bisection**.
4. Without loss of generality, assume that \(|V| = 2 \cdot n\). Why?
Theorem

**Bisection-Width is NP-complete.**

Proof

1. **Bisection-Width** is clearly in **NP**.
2. We reduce **Max-Bisection** to **Bisection-Width**.
3. Let \( (G = \langle V, E \rangle, K) \) denote an instance of **Max-Bisection**.
4. Without loss of generality, assume that \( |V| = 2 \cdot n \). Why?
5. The corresponding instance of **Bisection-Width** is: \( (G^c = \langle V, E^c \rangle, n^2 - K) \).
Theorem

**Bisection-Width is NP-complete.**

Proof

1. **Bisection-Width** is clearly in **NP**.
2. We reduce **Max-Bisection** to **Bisection-Width**.
3. Let \((G = \langle V, E \rangle, K)\) denote an instance of **Max-Bisection**.
4. Without loss of generality, assume that \(|V| = 2 \cdot n\). Why?
5. The corresponding instance of **Bisection-Width** is: \((G^c = \langle V, E^c \rangle, n^2 - K)\).
6. It is not hard to see that \(G\) has a bisection of size \(K\) or more if and only if \(G^c\) has a bisection of size \(n^2 - K\) or less.
Subset-Sum

Definition

Input: A list $S = \{a_1, a_2, \ldots, a_n\}$ and a target $T$.

Query: Is there a set $S' \subseteq S$, such that $\sum_{i \in S'} a_i = T$?
Subset-Sum

Definition

Input: A list \( S = \{a_1, a_2, \ldots, a_n\} \) and a target \( T \).

Query: Is there a set \( S' \subseteq S \) such that \( \sum_{i \in S'} a_i = T \)?
Definition

**Input:** A list $S = \{a_1, a_2, \ldots, a_n\}$ and a target $T$.  

Subset-Sum

**Definition**

**Input:** A list \( S = \{a_1, a_2, \ldots, a_n\} \) and a target \( T \).

**Query:** Is there a set \( S' \subseteq S \), such that \( \sum_{a_i \in S'} a_i = T \)?
Theorem

**UBSET-SUM is NP-complete.**

Proof

1. *UBSET-SUM is clearly in NP.*

2. We will reduce 3SAT to *UBSET-SUM.*

3. Given an instance \( \phi = C_1 \land C_2 \land \ldots \land C_m \) of \( m \) clauses over \( n \) variables, we construct the following instance of *UBSET-SUM:

   1. We will create \( 2 \cdot (m + n) \) numbers, each having \( (m + n) \) digits.

   2. Corresponding to each variable \( x_i \), there are two numbers \( T_i \) and \( F_i \).

   3. Corresponding to each clause \( C_i \), there are two rows \( S_{i1} \) and \( S_{i2} \).

   4. Finally, we create a target which has 1 in the first \( n \) digits and 4 in the final \( m \) digits.
Theorem

1. Subset-Sum is clearly in NP.
2. We will reduce 3SAT to Subset-Sum.
3. Given an instance \( \phi = C_1 \land C_2 \land \ldots \land C_m \) of \( m \) clauses over \( n \) variables, we construct the following instance of Subset-Sum:
   1. We will create \( 2 \cdot (m + n) \) numbers, each having \( (m + n) \) digits.
   2. Corresponding to each variable \( x_i \), there are two numbers \( T_i \) and \( F_i \).
   3. Corresponding to each clause \( C_i \), there are two rows \( S_{1i} \) and \( S_{2i} \).
   4. Finally, we create a target which has 1 in the first \( n \) digits and 4 in the final \( m \) digits.
Theorem

SUBSET-SUM is NP-complete.
Theorem

SUBSET-SUM is \textbf{NP-complete}.

Proof
**Theorem**

**SUBSET-SUM is NP-complete.**

**Proof**

1. **SUBSET-SUM is clearly in NP.**
Theorem

**SUBSET-SUM is NP-complete.**

Proof

1. **SUBSET-SUM** is clearly in **NP**.
2. We will reduce 3SAT to **SUBSET-SUM**.
Theorem

**SUBSET-SUM** is NP-complete.

Proof

1. **SUBSET-SUM** is clearly in NP.
2. We will reduce 3SAT to **SUBSET-SUM**.
3. Given an instance \( \phi = C_1 \land C_2 \land \ldots \land C_m \) of \( m \) clauses over \( n \) variables, we construct the following instance of **SUBSET-SUM**:
Theorem

**SUBSET-SUM** is **NP-complete**.

Proof

1. **SUBSET-SUM** is clearly in **NP**.
2. We will reduce 3SAT to **SUBSET-SUM**.
3. Given an instance \( \phi = C_1 \land C_2 \land \ldots \land C_m \) of \( m \) clauses over \( n \) variables, we construct the following instance of **SUBSET-SUM**:
   1. We will create \( 2 \cdot (m + n) \) numbers, each having \( (m + n) \) digits.
Theorem

**SUBSET-SUM** *is NP-complete.*

Proof

1. **SUBSET-SUM** is clearly in **NP**.
2. We will reduce 3SAT to **SUBSET-SUM**.
3. Given an instance \( \phi = C_1 \land C_2 \land \ldots \land C_m \) of \( m \) clauses over \( n \) variables, we construct the following instance of **SUBSET-SUM**:
   1. We will create \( 2 \cdot (m + n) \) numbers, each having \( (m + n) \) digits.
   2. Corresponding to each variable \( x_i \), there are two numbers \( T_i \) and \( F_i \).
Theorem

**SUBSET-SUM is NP-complete.**

Proof

1. **SUBSET-SUM** is clearly in **NP**.
2. We will reduce 3SAT to **SUBSET-SUM**.
3. Given an instance \( \phi = C_1 \land C_2 \land \ldots \land C_m \) of \( m \) clauses over \( n \) variables, we construct the following instance of **SUBSET-SUM**:
   1. We will create \( 2 \cdot (m + n) \) numbers, each having \( (m + n) \) digits.
   2. Corresponding to each variable \( x_i \), there are two numbers \( T_i \) and \( F_i \).
   3. Corresponding to each clause \( C_i \), there are two rows \( S_{l_1} \) and \( S_{l_2} \).
**Theorem**

*SUBSET-SUM is NP-complete.*

**Proof**

1. *SUBSET-SUM* is clearly in *NP*.
2. We will reduce 3SAT to *SUBSET-SUM*.
3. Given an instance $\phi = C_1 \land C_2 \land \ldots \land C_m$ of $m$ clauses over $n$ variables, we construct the following instance of *SUBSET-SUM*:
   1. We will create $2 \cdot (m + n)$ numbers, each having $(m + n)$ digits.
   2. Corresponding to each variable $x_i$, there are two numbers $T_i$ and $F_i$.
   3. Corresponding to each clause $C_i$, there are two rows $S_{i1}$ and $S_{i2}$.
   4. Finally, we create a target which has 1 in the first $n$ digits and 4 in the final $m$ digits.
Example

Let $\phi = (x_1, \neg x_3, \neg x_4) \land (\neg x_1, x_2, \neg x_4)$.

The corresponding instance of $\textsc{SubsetSum}$ is given below:

<table>
<thead>
<tr>
<th></th>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$x_4$</th>
<th>$c_1$</th>
<th>$c_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_1$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$F_1$</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$T_2$</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$F_2$</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$T_3$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$F_3$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$T_4$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$F_4$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

$S_1$: 1 0 0 0 0 1 0

$S_2$: 0 0 0 0 2 0

Target: 1 1 1 1 4 4

NP-completeness

Computational Complexity
Example

Let $\phi = (x_1, \neg x_3, \neg x_4) \land (\neg x_1, x_2, \neg x_4)$. The corresponding instance of $\text{SUBSET-SUM}$ is given below:

<table>
<thead>
<tr>
<th></th>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$x_4$</th>
<th>$c_1$</th>
<th>$c_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_1$</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$F_1$</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$T_2$</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$F_2$</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$T_3$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$F_3$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$T_4$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$F_4$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

$S_1 = [1, 0, 0, 0, 0, 1, 0]$  
$S_2 = [0, 0, 0, 0, 2, 0]$  
$T = [1, 1, 1, 1, 4, 4]$
Example

Let $\phi = (x_1, \neg x_3, \neg x_4) \land (\neg x_1, x_2, \neg x_4)$. 
Example

Let $\phi = (x_1, \neg x_3, \neg x_4) \land (\neg x_1, x_2, \neg x_4)$.

The corresponding instance of SUBSET-SUM is given below:
Example

Let $\phi = (x_1, \neg x_3, \neg x_4) \land (\neg x_1, x_2, \neg x_4)$.

The corresponding instance of SUBSET-SUM is given below:

<table>
<thead>
<tr>
<th></th>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$x_4$</th>
<th>$c_1$</th>
<th>$c_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_1$</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$F_1$</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$T_2$</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$F_2$</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$T_3$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$F_3$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$T_4$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$F_4$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$S1_1$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$S1_2$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>$S2_1$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$S2_2$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td><strong>Target</strong></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>
Assume that $\phi$ is satisfiable.

Pick all the rows that correspond to true literals.

Since the assignment is consistent, the first $n$ bits of the target $T$ are met by these $n$ literals.

Since each clause $C_i$ is satisfied, at least one number in which $c_i = 1$ is picked.

Depending on whether $C_i$ is satisfied by one literal, two literals or all three literals, we pick $S_1$ and $S_2$, or $S_2$ or $S_1$ respectively.

Clearly the final $m$ bits of the target are met.

Now assume that the target $T$ is met by some subset of numbers.

We must have picked exactly one of $T_i$ and $F_i$ for each $i$. Why?

If $T_i$ is picked, set $x_i$ to true; otherwise, set it to false.

We thus have a consistent assignment.

Since the final $m$ bits of the target are met, we cannot have a case where all literals of a clause are set to false.
Assume that $\phi$ is satisfiable.

Pick all the rows that correspond to true literals.

Since the assignment is consistent, the first $n$ bits of the target $T$ are met by these $n$ literals.

Since each clause $C_i$ is satisfied, at least one number in which $c_i = 1$ is picked.

Depending on whether $C_i$ is satisfied by one literal, two literals or all three literals, we pick $SI_1$ and $SI_2$, or $SI_2$ or $SI_1$ respectively.

Clearly the final $m$ bits of the target are met.

Now assume that the target $T$ is met by some subset of numbers.

We must have picked exactly one of $T_i$ and $F_i$ for each $i$. Why?

If $T_i$ is picked, set $x_i$ to true; otherwise, set it to false.

We thus have a consistent assignment.

Since the final $m$ bits of the target are met, we cannot have a case where all literals of a clause are set to false.
Assume that $\phi$ is satisfiable.
### Proof

1. Assume that $\phi$ is satisfiable.
2. Pick all the rows that correspond to **true** literals.
Argument

**Proof**

1. Assume that $\phi$ is satisfiable.
2. Pick all the rows that correspond to true literals.
3. Since the assignment is consistent, the first $n$ bits of the target $T$ are met by these $n$ literals.
Proof

1. Assume that $\phi$ is satisfiable.
2. Pick all the rows that correspond to true literals.
3. Since the assignment is consistent, the first $n$ bits of the target $T$ are met by these $n$ literals.
4. Since each clause $C_i$ is satisfied, at least one number in which $c_i = 1$ is picked.
### Argument

<table>
<thead>
<tr>
<th>Proof</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Assume that $\phi$ is satisfiable.</td>
</tr>
<tr>
<td>2. Pick all the rows that correspond to <strong>true</strong> literals.</td>
</tr>
<tr>
<td>3. Since the assignment is consistent, the first $n$ bits of the target $T$ are met by these $n$ literals.</td>
</tr>
<tr>
<td>4. Since each clause $C_i$ is satisfied, at least one number in which $c_i = 1$ is picked.</td>
</tr>
<tr>
<td>5. Depending on whether $C_i$ is satisfied by one literal, two literals or all three literals, we pick $SI_1$ and $SI_2$, or $SI_2$ or $SI_1$ respectively.</td>
</tr>
</tbody>
</table>
### Proof

1. Assume that $\phi$ is satisfiable.
2. Pick all the rows that correspond to **true** literals.
3. Since the assignment is consistent, the first $n$ bits of the target $T$ are met by these $n$ literals.
4. Since each clause $C_i$ is satisfied, at least one number in which $c_i = 1$ is picked.
5. Depending on whether $C_i$ is satisfied by one literal, two literals or all three literals, we pick $S_{I_1}$ and $S_{I_2}$, or $S_{I_2}$ or $S_{I_1}$ respectively.
6. Clearly the final $m$ bits of the target are met.
Assume that $\phi$ is satisfiable.

Pick all the rows that correspond to true literals.

Since the assignment is consistent, the first $n$ bits of the target $T$ are met by these $n$ literals.

Since each clause $C_i$ is satisfied, at least one number in which $c_i = 1$ is picked.

Depending on whether $C_i$ is satisfied by one literal, two literals or all three literals, we pick $SI_1$ and $SI_2$, or $SI_2$ or $SI_1$ respectively.

Clearly the final $m$ bits of the target are met.

Now assume that the target $T$ is met by some subset of numbers.
Argument

Proof

1. Assume that $\phi$ is satisfiable.
2. Pick all the rows that correspond to true literals.
3. Since the assignment is consistent, the first $n$ bits of the target $T$ are met by these $n$ literals.
4. Since each clause $C_i$ is satisfied, at least one number in which $c_i = 1$ is picked.
5. Depending on whether $C_i$ is satisfied by one literal, two literals or all three literals, we pick $SI_1$ and $SI_2$, or $SI_2$ or $SI_1$ respectively.
6. Clearly the final $m$ bits of the target are met.
7. Now assume that the target $T$ is met by some subset of numbers.
8. We must have picked exactly one of $T_i$ and $F_i$ for each $i$. Why?
Argument

Proof

1. Assume that $\phi$ is satisfiable.
2. Pick all the rows that correspond to true literals.
3. Since the assignment is consistent, the first $n$ bits of the target $T$ are met by these $n$ literals.
4. Since each clause $C_i$ is satisfied, at least one number in which $c_i = 1$ is picked.
5. Depending on whether $C_i$ is satisfied by one literal, two literals or all three literals, we pick $SI_1$ and $SI_2$, or $SI_2$ or $SI_1$ respectively.
6. Clearly the final $m$ bits of the target are met.
7. Now assume that the target $T$ is met by some subset of numbers.
8. We must have picked exactly one of $T_i$ and $F_i$ for each $i$. Why?
9. If $T_i$ is picked, set $x_i$ to true; otherwise, set it to false.
Argument

Proof

1. Assume that \( \phi \) is satisfiable.
2. Pick all the rows that correspond to \texttt{true} literals.
3. Since the assignment is consistent, the first \( n \) bits of the target \( T \) are met by these \( n \) literals.
4. Since each clause \( C_i \) is satisfied, at least one number in which \( c_i = 1 \) is picked.
5. Depending on whether \( C_i \) is satisfied by one literal, two literals or all three literals, we pick \( S_{I_1} \) and \( S_{I_2} \), or \( S_{I_2} \) or \( S_{I_1} \) respectively.
6. Clearly the final \( m \) bits of the target are met.
7. Now assume that the target \( T \) is met by some subset of numbers.
8. We must have picked exactly one of \( T_i \) and \( F_i \) for each \( i \). Why?
9. If \( T_i \) is picked, set \( x_i \) to \texttt{true}; otherwise, set it to \texttt{false}.
10. We thus have a consistent assignment.
Proof

1. Assume that $\phi$ is satisfiable.
2. Pick all the rows that correspond to true literals.
3. Since the assignment is consistent, the first $n$ bits of the target $T$ are met by these $n$ literals.
4. Since each clause $C_i$ is satisfied, at least one number in which $c_i = 1$ is picked.
5. Depending on whether $C_i$ is satisfied by one literal, two literals or all three literals, we pick $SI_1$ and $SI_2$, or $SI_2$ or $SI_1$ respectively.
6. Clearly the final $m$ bits of the target are met.
7. Now assume that the target $T$ is met by some subset of numbers.
8. We must have picked exactly one of $T_i$ and $F_i$ for each $i$. Why?
9. If $T_i$ is picked, set $x_i$ to true; otherwise, set it to false.
10. We thus have a consistent assignment.
11. Since the final $m$ bits of the target are met, we cannot have a case where all literals of a clause are set to false.
Partition

**Definition**

Input: A list of numbers $S = \{a_1, a_2, \ldots, a_n\}$.

Query: Is there a set $S' \subseteq S$ such that $\sum a_j \in S' a_j = \sum a_j \notin S' a_j$?
Definition

**Input:** A list of numbers $S = \{a_1, a_2, \ldots a_n\}$. 
Partition

Definition

**Input:** A list of numbers $S = \{a_1, a_2, \ldots a_n\}$.

**Query:** Is there a set $S' \subseteq S$, such that $\sum_{a_j \in S'} a_j = \sum_{a_j \in S - S'} a_j$?
**Theorem**

**PARTITION** is NP-complete.

**Proof**

1. **PARTITION** is clearly in NP.

2. We reduce **SUBSET-SUM** to **PARTITION**.

3. Let \((S = \{a_1, a_2, \ldots, a_n\}, T)\) denote an instance of **SUBSET-SUM**.

4. The corresponding instance of **PARTITION** is:

\[
R = \{a_1, a_2, \ldots, a_n, L + T, 2 \cdot L - T\},
\]

where \(L = \sum_{i \in S} a_i\).
Theorem

**PARTITION** is NP-complete.

Proof

1. **PARTITION** is clearly in NP.
2. We reduce **SUBSET-SUM** to **PARTITION**.
3. Let \((S = \{a_1, a_2, \ldots, a_n\}, T)\) denote an instance of **SUBSET-SUM**.
4. The corresponding instance of **PARTITION** is:
   \[ R = \{a_1, a_2, \ldots, a_n, L + T, 2 \cdot L - T\} \]
   where \(L = \sum_{i \in S} a_i\).
**Theorem**

**PARTITION** is **NP-complete**.
Theorem

**PARTITION is NP-complete.**

Proof
Theorem

\textsc{Partition} is \textbf{NP-complete}.

Proof

1. \textsc{Partition} is clearly in \textbf{NP}.
Theorem

**PARTITION** is **NP-complete**.

Proof

1. **PARTITION** is clearly in **NP**.
2. We reduce **SUBSET-SUM to PARTITION**.
Theorem

**PARTITION is NP-complete.**

Proof

1. **PARTITION** is clearly in **NP**.
2. We reduce **SUBSET-SUM** to **PARTITION**.
3. Let \((S = \{a_1, a_2, \ldots, a_n\}, T)\) denote an instance of **SUBSET-SUM**.
Theorem

**PARTITION is NP-complete.**

Proof

1. **PARTITION** is clearly in **NP**.
2. We reduce **SUBSET-SUM** to **PARTITION**.
3. Let \((S = \{a_1, a_2, \ldots, a_n\}, T)\) denote an instance of **SUBSET-SUM**.
4. The corresponding instance of **PARTITION** is:
**Theorem**

\[ \text{PARTITION is NP-complete.} \]

**Proof**

1. \( \text{PARTITION} \) is clearly in \( \text{NP} \).
2. We reduce \( \text{SUBSET-SUM} \) to \( \text{PARTITION} \).
3. Let \( (S = \{a_1, a_2, \ldots, a_n\}, T) \) denote an instance of \( \text{SUBSET-SUM} \).
4. The corresponding instance of \( \text{PARTITION} \) is:

\[ R = \{a_1, a_2, \ldots, a_n, L + T, 2 \cdot L - T\}, \text{ where } L = \sum_{a_i \in S} a_i. \]
Argument

1. Assume that $S$ has a subset $S'$ which sums to $T$.

2. We can partition the set $R$ into the sets $S' \cup \{2 \cdot L - T\}$ and $S \setminus S' \cup \{L + T\}$.

3. Both sets sum to $2 \cdot L$.

4. Now assume that $R$ has a partition $(R_1, R_2)$.

5. Both $R_1$ and $R_2$ sum to $2 \cdot L$.

6. Can $L + T$ and $2 \cdot L - T$ belong to the same partition?

7. Assume that $2 \cdot L - T \in R_1$.

8. The remaining elements in $R_1$ are all in $S$ and clearly sum to $T$. 

NP-completeness
Assume that $S$ has a subset $S'$ which sums to $T$.

We can partition the set $R$ into the sets $S' \cup \{2 \cdot L - T\}$ and $S \setminus S' \cup \{L + T\}$.

Both sets sum to $2 \cdot L$.

Can $L + T$ and $2 \cdot L - T$ belong to the same partition?

Assume that $2 \cdot L - T \in R_1$.

The remaining elements in $R_1$ are all in $S$ and clearly sum to $T$.
Completing the argument

1. Assume that \( S \) has a subset \( S' \) which sums to \( T \).
Completing the argument

1. Assume that $S$ has a subset $S'$ which sums to $T$.
2. We can partition the set $R$ into the sets $S' \cup \{2 \cdot L - T\}$ and $S \setminus S' \cup \{L + T\}$.
Completing the argument

1. Assume that $S$ has a subset $S'$ which sums to $T$.
2. We can partition the set $R$ into the sets $S' \cup \{2 \cdot L - T\}$ and $S \setminus S' \cup \{L + T\}$.
3. Both sets sum to
<table>
<thead>
<tr>
<th>Argument</th>
</tr>
</thead>
<tbody>
<tr>
<td>Completing the argument</td>
</tr>
<tr>
<td>1. Assume that $S$ has a subset $S'$ which sums to $T$.</td>
</tr>
<tr>
<td>2. We can partition the set $R$ into the sets $S' \cup {2 \cdot L - T}$ and $S \setminus S' \cup {L + T}$.</td>
</tr>
<tr>
<td>3. Both sets sum to $2 \cdot L!$.</td>
</tr>
</tbody>
</table>
Completing the argument

1. Assume that $S$ has a subset $S'$ which sums to $T$.
2. We can partition the set $R$ into the sets $S' \cup \{2 \cdot L - T\}$ and $S \setminus S' \cup \{L + T\}$.
3. Both sets sum to $2 \cdot L$!
4. Now assume that $R$ has a partition $(R_1, R_2)$. 
Completing the argument

1. Assume that $S$ has a subset $S'$ which sums to $T$.
2. We can partition the set $R$ into the sets $S' \cup \{2 \cdot L - T\}$ and $S \setminus S' \cup \{L + T\}$.
3. Both sets sum to $2 \cdot L$!
4. Now assume that $R$ has a partition $(R_1, R_2)$.
5. Both $R_1$ and $R_2$ sum to
Completing the argument

1. Assume that $S$ has a subset $S'$ which sums to $T$.
2. We can partition the set $R$ into the sets $S' \cup \{2 \cdot L - T\}$ and $S \setminus S' \cup \{L + T\}$.
3. Both sets sum to $2 \cdot L$!
4. Now assume that $R$ has a partition $(R_1, R_2)$.
5. Both $R_1$ and $R_2$ sum to $2 \cdot L$. 
Completing the argument

1. Assume that $S$ has a subset $S'$ which sums to $T$.
2. We can partition the set $R$ into the sets $S' \cup \{2 \cdot L - T\}$ and $S \setminus S' \cup \{L + T\}$.
3. Both sets sum to $2 \cdot L$.
4. Now assume that $R$ has a partition $(R_1, R_2)$.
5. Both $R_1$ and $R_2$ sum to $2 \cdot L$.
6. Can $L + T$ and $2 \cdot L - T$ belong to the same partition?
Assume that $S$ has a subset $S'$ which sums to $T$.

We can partition the set $R$ into the sets $S' \cup \{2 \cdot L - T\}$ and $S \setminus S' \cup \{L + T\}$.

Both sets sum to $2 \cdot L$!

Now assume that $R$ has a partition $(R_1, R_2)$.

Both $R_1$ and $R_2$ sum to $2 \cdot L$.

Can $L + T$ and $2 \cdot L - T$ belong to the same partition?

Assume that $2 \cdot L - T \in R_1$. 

Assume that $2 \cdot L - T \in R_1$. 

The remaining elements in $R_1$ are all in $S$ and clearly sum to $T$!
Argument

Completing the argument

1. Assume that $S$ has a subset $S'$ which sums to $T$.
2. We can partition the set $R$ into the sets $S' \cup \{2 \cdot L - T\}$ and $S \setminus S' \cup \{L + T\}$.
3. Both sets sum to $2 \cdot L$.
4. Now assume that $R$ has a partition $(R_1, R_2)$.
5. Both $R_1$ and $R_2$ sum to $2 \cdot L$.
6. Can $L + T$ and $2 \cdot L - T$ belong to the same partition?
7. Assume that $2 \cdot L - T \in R_1$.
8. The remaining elements in $R_1$ are all in $S$ and clearly sum to $T$!
Knapsack

**Definition**

**Input:** Vectors

\[ p = (p_1, p_2, ..., p_n), \]

\[ w = (w_1, w_2, ..., w_n), \]

integers \( P \) and \( W \).

**Query:** Is there an \( x = [x_1, x_2, ..., x_n] \) \( \in \{0, 1\}^n \) such that

\[ \sum_{i=1}^{n} w_i \cdot x_i \leq W \]

\[ \sum_{i=1}^{n} p_i \cdot x_i \geq P \,? \]
Knapsack

**Definition**

Input: Vectors $p = (p_1, p_2, \ldots, p_n)$, $w = (w_1, w_2, \ldots, w_n)$, integers $P$ and $W$.

Query: Is there an $x = [x_1, x_2, \ldots, x_n] \in \{0, 1\}^n$ such that $\sum_{i=1}^{n} w_i \cdot x_i \leq W$ and $\sum_{i=1}^{n} p_i \cdot x_i \geq P$?

NP-completeness

Computational Complexity
Knapsack

**Definition**

**Input:** Vectors \( \mathbf{p} = (p_1, p_2, \ldots, p_n) \), \( \mathbf{w} = (w_1, w_2, \ldots w_n) \), integers \( P \) and \( W \).
**Definition**

**Input:** Vectors \( p = (p_1, p_2, \ldots, p_n) \), \( w = (w_1, w_2, \ldots, w_n) \), integers \( P \) and \( W \).

**Query:** Is there an \( x = [x_1, x_2, \ldots, x_n] \in \{0, 1\}^n \) such that
**Knapsack**

**Definition**

**Input:** Vectors \( p = (p_1, p_2, \ldots, p_n) \), \( w = (w_1, w_2, \ldots w_n) \), integers \( P \) and \( W \).

**Query:** Is there an \( x = [x_1, x_2, \ldots x_n] \in \{0, 1\}^n \) such that
Definition

**Input:** Vectors $p = (p_1, p_2, \ldots, p_n)$, $w = (w_1, w_2, \ldots, w_n)$, integers $P$ and $W$.

**Query:** Is there an $x = [x_1, x_2, \ldots, x_n] \in \{0, 1\}^n$ such that

$$\sum_{i=1}^{n} w_i \cdot x_i \leq W$$
**Definition**

**Input:** Vectors $\mathbf{p} = (p_1, p_2, \ldots, p_n)$, $\mathbf{w} = (w_1, w_2, \ldots, w_n)$, integers $P$ and $W$.

**Query:** Is there an $\mathbf{x} = [x_1, x_2, \ldots, x_n] \in \{0, 1\}^n$ such that

$$\sum_{i=1}^{n} w_i \cdot x_i \leq W$$

$$\sum_{i=1}^{n} p_i \cdot x_i \geq P?$$
Theorem

\[ \text{KNAPSACK} \text{ is NP-complete.} \]

Proof

1. \text{KNAPSACK} is clearly in \text{NP}.

2. We reduce \text{SUBSET-SUM} to \text{KNAPSACK}.

3. Given an instance of \text{SUBSET-SUM}, create the following instance of \text{KNAPSACK}:

   1. Set \(w_i = p_i = a_i\), \(\forall i = 1, 2, \ldots, n\).
   
   2. Set \(W = P = T\).

Can you establish that the instance of \text{SUBSET-SUM} is true if and only if the instance of \text{KNAPSACK} is?
**Theorem**

**K-NAPACK** is clearly in **NP**. We reduce **S-UBSET-SUM** to **K-NAPACK**.

Given an instance of **S-UBSET-SUM**, create the following instance of **K-NAPACK**:

1. Set \( w_i = p_i = a_i \), for all \( i = 1, 2, \ldots, n \).
2. Set \( W = P = T \).

Can you establish that the instance of **S-UBSET-SUM** is true if and only if the instance of **K-NAPACK** is?
Theorem

**KNAPSACK** is **NP-complete**.
Theorem

**KNAPSACK is NP-complete.**

Proof
Theorem

**KNAPSACK** is **NP-complete**.

Proof

1. **KNAPSACK** is clearly in **NP**.
Theorem

**KNAPSACK is NP-complete.**

Proof

1. **KNAPSACK** is clearly in **NP**.
2. We reduce **SUBSET-SUM** to **KNAPSACK**.
Theorem

Knapsack is NP-complete.

Proof

1. Knapsack is clearly in NP.
2. We reduce Subset-Sum to Knapsack.
3. Given an instance of Subset-Sum, create the following instance of Knapsack:
Theorem

**Knapsack** is **NP-complete**.

Proof

1. **Knapsack** is clearly in **NP**.
2. We reduce **Subset-Sum** to **Knapsack**.
3. Given an instance of **Subset-Sum**, create the following instance of **Knapsack**:
   1. Set \( w_i = p_i = a_i \), \( \forall i = 1, 2, \ldots n \).
Theorem

**K NAPSACK** _is NP-complete_.

Proof

1. **K NAPSACK** is clearly in **NP**.
2. We reduce **SUBSET-SUM** to **K NAPSACK**.
3. Given an instance of **SUBSET-SUM**, create the following instance of **K NAPSACK**:
   1. Set \( w_i = p_i = a_i, \forall i = 1, 2, \ldots n \).
   2. Set \( W = P = T \).
**Theorem**

\textbf{KNAPSACK} \textit{is NP-complete}.

**Proof**

1. \textbf{KNAPSACK} is clearly in \textbf{NP}.
2. We reduce \textsc{Subset-Sum} to \textbf{KNAPSACK}.
3. Given an instance of \textsc{Subset-Sum}, create the following instance of \textbf{KNAPSACK}:
   1. Set $w_i = p_i = a_i, \forall i = 1, 2, \ldots n$.
   2. Set $W = P = T$.
   3. Can you establish that the instance of \textsc{Subset-Sum} is \textit{true} if and only if the instance of \textbf{KNAPSACK} is?
The Power of Integer Programming

Exercise

Reduce all the problems discussed thus far to Integer Programming.
The Power of Integer Programming

Exercise

Reduce all the problems discussed thus far to Integer Programming.
Directed Hamilton Path

**Definition**

Input: A directed graph $G = \langle V, E \rangle$.

Query: Is there a dipath in $G$ that touches every vertex exactly once. Such a path if it exists, is called a Directed Hamilton Path.

**Reduction**

$3\text{SAT} \leq \text{DIRECTED-HAMILTON-PATH}$. 

**NP-completeness**

Computational Complexity
Directed Hamilton Path

Definition

Input: A directed graph $G = (V, E)$. 

Query: Is there a dipath in $G$ that touches every vertex exactly once. Such a path if it exists, is called a Directed Hamilton Path.
Directed Hamilton Path

Definition

**Input:** A directed graph \( G = \langle V, E \rangle \).

**Query:** Is there a dipath in \( G \) that touches every vertex exactly once.
Definition

**Input:** A directed graph $G = (V, E)$.

**Query:** Is there a dipath in $G$ that touches every vertex exactly once.

Such a path if it exists, is called a Directed Hamilton Path.
Definition

**Input:** A directed graph $G = \langle V, E \rangle$.

**Query:** Is there a dipath in $G$ that touches every vertex exactly once.

Such a path if it exists, is called a Directed Hamilton Path.

Reduction
Definition

**Input:** A directed graph $G = \langle V, E \rangle$.

**Query:** Is there a dipath in $G$ that touches every vertex exactly once.

Such a path if it exists, is called a Directed Hamilton Path.

Reduction

$3\text{SAT} \leq \text{DIRECTED-HAMPATH}$. 
Definition
Input: A directed graph $G = \langle V, E \rangle$ and two vertices $s, t \in V$.
Query: Is there a dipath from $s$ to $t$ in $G$ that touches all the vertices in $V - \{s, t\}$ exactly once?
Such a path if it exists, is called an $s - t$ Directed Hamilton Path.

Reduction
Same as above.

NP-completeness
Computational Complexity
Definition

Input: A directed graph $G = \langle V, E \rangle$ and two vertices $s, t \in V$.

Query: Is there a dipath from $s$ to $t$ in $G$ that touches all the vertices in $V - \{s, t\}$ exactly once?

Such a path if it exists, is called an $s-t$ Directed Hamilton Path.
**Definition**

**Input:** A directed graph $G = (V, E)$ and two vertices $s, t \in V$. 

**Query:** Is there a dipath from $s$ to $t$ in $G$ that touches all the vertices in $V - \{s, t\}$ exactly once? Such a path if it exists, is called an $s-t$ Directed Hamilton Path.
Definition

**Input:** A directed graph $G = \langle V, E \rangle$ and two vertices $s, t \in V$.

**Query:** Is there a dipath from $s$ to $t$ in $G$ that touches all the vertices in $V - \{s, t\}$ exactly once?
s – t Directed Hamilton Path

Definition

**Input:** A directed graph $G = (V, E)$ and two vertices $s, t \in V$.

**Query:** Is there a dipath from $s$ to $t$ in $G$ that touches all the vertices in $V - \{s, t\}$ exactly once?

Such a path if it exists, is called an $s – t$ Directed Hamilton Path.
**Definition**

**Input:** A directed graph $G = (V, E)$ and two vertices $s, t \in V$.

**Query:** Is there a dipath from $s$ to $t$ in $G$ that touches all the vertices in $V - \{s, t\}$ exactly once?

Such a path if it exists, is called an $s - t$ Directed Hamilton Path.

**Reduction**

---

**NP-completeness**

**Computational Complexity**
**Definition**

**Input:** A directed graph $G = (V, E)$ and two vertices $s, t \in V$.

**Query:** Is there a dipath from $s$ to $t$ in $G$ that touches all the vertices in $V - \{s, t\}$ exactly once?

Such a path if it exists, is called an $s - t$ Directed Hamilton Path.

**Reduction**

Same as above.
Definition
Input: A directed graph $G = \langle V, E \rangle$.
Query: Is there a directed cycle in $G$, that goes through each vertex exactly once? Such a cycle if it exists, is called a Directed Hamilton Circuit or Directed Hamilton Cycle.

Reduction
$\text{DIRECTED PATH} \leq \text{DIRECTED CYCLE}$. 

Exercise
Can you provide a reduction from $\text{DIRECTED PATH}$ to $\text{DIRECTED CYCLE}$?
**Definition**

**Input:** A directed graph \( G = (V, E) \).
Definition

**Input:** A directed graph $G = (V, E)$.

**Query:** Is there a directed cycle in $G$, that goes through each vertex exactly once?
**Definition**

**Input:** A directed graph $G = (V, E)$.

**Query:** Is there a directed cycle in $G$, that goes through each vertex exactly once? Such a cycle if it exists, is called a Directed Hamilton Circuit or Directed Hamilton Cycle.
Definition

**Input:** A directed graph $G = (V, E)$.

**Query:** Is there a directed cycle in $G$, that goes through each vertex exactly once?

Such a cycle if it exists, is called a Directed Hamilton Circuit or Directed Hamilton Cycle.

Reduction

Exercise: Can you provide a reduction from **DIRECTED PATH** to **DIRECTED CYCLE**?
Directed Hamilton Circuit

**Definition**

**Input:** A directed graph $G = (V, E)$.

**Query:** Is there a directed cycle in $G$, that goes through each vertex exactly once?

Such a cycle if it exists, is called a Directed Hamilton Circuit or Directed Hamilton Cycle.

**Reduction**

$s \to t$ \textsc{Directed-HamPath} $\leq$ \textsc{Directed-HamCycle}.
Directed Hamilton Circuit

**Definition**

**Input:** A directed graph $G = (V, E)$.

**Query:** Is there a directed cycle in $G$, that goes through each vertex exactly once?

Such a cycle if it exists, is called a Directed Hamilton Circuit or Directed Hamilton Cycle.

**Reduction**

$s - t$ \textsc{Directed-HamPath} \leq \textsc{Directed-HamCycle}.

**Exercise**
### Directed Hamilton Circuit

**Definition**

**Input:** A directed graph $G = \langle V, E \rangle$.

**Query:** Is there a directed cycle in $G$, that goes through each vertex exactly once?

Such a cycle if it exists, is called a Directed Hamilton Circuit or Directed Hamilton Cycle.

**Reduction**

$s \rightarrow t$ $\text{DIRECTED-HAMPATH} \leq \text{DIRECTED-HAMCYCLE}$.

**Exercise**

Can you provide a reduction from $\text{DIRECTED-HAMPATH}$ to $\text{DIRECTED-HAMCYCLE}$?
Undirected Hamilton Cycle

**Definition**

Input: An undirected graph $G = \langle V, E \rangle$.

Query: Is there an undirected Hamilton cycle in $G$?

**Reduction**

$\text{DIRECTED-HAM-CYCLE} \leq \text{HAM-CYCLE}$.
Undirected Hamilton Cycle

**Definition**

**Input**: An undirected graph $G = \langle V, E \rangle$. 
Undirected Hamilton Cycle

**Definition**

**Input:** An undirected graph $G = \langle V, E \rangle$.

**Query:** Is there an undirected Hamilton cycle in $G$?
Undirected Hamilton Cycle

**Definition**

**Input:** An undirected graph $G = (V, E)$.  

**Query:** Is there an undirected Hamilton cycle in $G$?

**Reduction**

$D\text{IRECTED-HAM}\text{CYCLE} \leq \text{HAMILTON-CYCLE}$.  

NP-completeness
Undirected Hamilton Cycle

**Definition**

*Input*: An undirected graph $G = \langle V, E \rangle$.

*Query*: Is there an undirected Hamilton cycle in $G$?

**Reduction**

$\text{DIRECTED-HAMCYCLE} \leq \text{HAMCYCLE}$. 

NP-completeness
Traveling Salesman Problem

Definition

Input: A directed graph $G = \langle V, E \rangle$, a pairwise distance matrix $D$ and a budget $B$.

Query: Is there a Hamilton cycle in $G$ with cost at most $B$?

Reduction

$D\text{IRECTED-HAMCYCLE} \leq \text{TSP}(D)$. 

NP-completeness

Computational Complexity
Traveling Salesman Problem

Definition

**Input:** An directed graph \( G = \langle V, E \rangle \), a pairwise distance matrix \( D \) and a budget \( B \).

Query: Is there a Hamilton cycle in \( G \) with cost at most \( B \)?

Reduction \( D\text{IRECTED-HAMCYCLE} \leq \text{TSP}(D) \).

NP-completeness

Computational Complexity
Traveling Salesman Problem

**Definition**

**Input:** An directed graph $G = \langle V, E \rangle$, a pairwise distance matrix $D$ and a budget $B$.

**Query:** Is there a Hamilton cycle in $G$ with cost at most $B$?
Traveling Salesman Problem

Definition

**Input:** An directed graph $G = \langle V, E \rangle$, a pairwise distance matrix $D$ and a budget $B$.

**Query:** Is there a Hamilton cycle in $G$ with cost at most $B$?

Reduction

NP-completeness
Traveling Salesman Problem

**Definition**

*Input:* An directed graph $G = \langle V, E \rangle$, a pairwise distance matrix $D$ and a budget $B$.

*Query:* Is there a Hamilton cycle in $G$ with cost at most $B$?

**Reduction**

$\text{DIRECTED-HAMCYCLE} \leq \text{TSP}(D)$. 

**NP-completeness**

Computational Complexity
Optimization Problems on Graphs
Number Problems
The Power of Integer Programming
Paths, trees and Circuits

Traveling Salesman Problem (Triangle Inequality)

**Definition**

**Input:**
An directed graph $G = \langle V, E \rangle$, a pairwise distance matrix $D$ and a budget $B$.

It is assumed that the distance matrix $D$ enjoys the following property (known as triangle inequality):

$$d(u, v) \leq d(u, w) + d(w, v), \forall u, v, w \in V$$

**Query:**
Is there a Hamilton cycle in $G$ with cost at most $B$?

**Reduction**

$DRECTED$-CYCLE $\leq_{\triangle}$ TSP($D$).

**NP-completeness**

**Computational Complexity**
Definition

**Input:** An directed graph $G = \langle V, E \rangle$, a pairwise distance matrix $D$ and a budget $B$. 

It is assumed that the distance matrix $D$ enjoys the following property (known as triangle inequality):

$$d(u, v) \leq d(u, w) + d(w, v), \quad \forall u, v, w \in V$$

Query: Is there a Hamilton cycle in $G$ with cost at most $B$?

Reduction $\text{DIRECTED-HAMCYCLE} \leq \Delta \text{TSP}(D)$. 

NP-completeness

Computational Complexity
Definition

**Input:** An directed graph $G = \langle V, E \rangle$, a pairwise distance matrix $D$ and a budget $B$.

It is assumed that the distance matrix $D$ enjoys the following property (known as triangle inequality):

$$d(u, v) \leq d(u, w) + d(w, v), \quad \forall u, v, w \in V$$
Definition

**Input:** An directed graph $G = \langle V, E \rangle$, a pairwise distance matrix $D$ and a budget $B$.

It is assumed that the distance matrix $D$ enjoys the following property (known as triangle inequality):

$$d(u, v) \leq d(u, w) + d(w, v), \ \forall u, v, w \in V$$
Traveling Salesman Problem (Triangle Inequality)

**Definition**

**Input:** An directed graph $G = \langle V, E \rangle$, a pairwise distance matrix $D$ and a budget $B$.

It is assumed that the distance matrix $D$ enjoys the following property (known as triangle inequality):

$$d(u, v) \leq d(u, w) + d(w, v), \forall u, v, w \in V$$

**Query:** Is there a Hamilton cycle in $G$ with cost at most $B$?
Definition

Input: An directed graph $G = \langle V, E \rangle$, a pairwise distance matrix $D$ and a budget $B$.

It is assumed that the distance matrix $D$ enjoys the following property (known as triangle inequality):

$$d(u, v) \leq d(u, w) + d(w, v), \quad \forall u, v, w \in V$$

Query: Is there a Hamilton cycle in $G$ with cost at most $B$?

Reduction

DIRECTED-HAMCYCLE $\leq_{\Delta}$ TSP$(D)$. 
Traveling Salesman Problem (Triangle Inequality)

Definition

**Input:** An directed graph $G = \langle V, E \rangle$, a pairwise distance matrix $D$ and a budget $B$.

It is assumed that the distance matrix $D$ enjoys the following property (known as triangle inequality):

$$d(u, v) \leq d(u, w) + d(w, v), \quad \forall u, v, w \in V$$

**Query:** Is there a Hamilton cycle in $G$ with cost at most $B$?

Reduction

**DIRECTED-HAMCYCLE $\leq \triangle TSP(D)$**.
Optimization Problems on Graphs
Number Problems
The Power of Integer Programming
Paths, trees and Circuits

Longest Path

**Definition**

Input: An directed graph \( G = \langle V, E, c \rangle \), where \( c : E \to \mathbb{Z} \) is a cost function and a cost value \( K \).

Query: Is there a path in \( G \) of cost at least \( K \)?

**Reduction** \( \text{DIRECTED-HAM-PATH} \leq \text{LONGEST-PATH} \).
Longest Path

Definition

Input: An directed graph $G = \langle V, E, c \rangle$, where $c: E \rightarrow \mathbb{Z}$ is a cost function and a cost value $K$.

Query: Is there a path in $G$ of cost at least $K$?

Reduction $D_{\text{DIRECTED-HAMPATH}} \leq L_{\text{LONGEST-PATH}}$. 

NP-completeness

Computational Complexity
Definition

**Input:** An directed graph $G = \langle V, E, c \rangle$, where $c : E \rightarrow \mathbb{Z}$ is a cost function and a cost value $K$. 
Longest Path

Definition

**Input:** An directed graph \( G = \langle V, E, c \rangle \), where \( c : E \to \mathbb{Z} \) is a cost function and a cost value \( K \).

**Query:** Is there a path in \( G \) of cost at least \( K \)?
**Definition**

**Input:** An directed graph \( G = \langle V, E, c \rangle \), where \( c : E \rightarrow \mathbb{Z} \) is a cost function and a cost value \( K \).

**Query:** Is there a path in \( G \) of cost at least \( K \)?

**Reduction**
Longest Path

Definition

Input: An directed graph $G = (V, E, c)$, where $c : E \rightarrow \mathbb{Z}$ is a cost function and a cost value $K$.

Query: Is there a path in $G$ of cost at least $K$?

Reduction

$\text{DIRECTED-HAMPATH} \leq \text{LONGEST-PATH}$. 
Longest Circuit

Definition

Input: An directed graph $G = \langle V, E, c \rangle$, where $c: E \rightarrow \mathbb{Z}$ is a cost function and a cost value $K$.

Query: Is there a cycle in $G$ of cost at least $K$?

Reduction $D\text{IRECTED-HAMCYCLE} \leq L\text{ONGEST-PATH}$. 

NP-completeness and Computational Complexity
Longest Circuit

Definition

Input: An directed graph $G = \langle V, E, c \rangle$, where $c: E \to \mathbb{Z}$ is a cost function and a cost value $K$.

Query: Is there a cycle in $G$ of cost at least $K$?

Reduction $\text{DIRECTED-HAMCYCLE} \leq \text{LONGEST-PATH}$. 

NP-completeness

Computational Complexity
Longest Circuit

Definition

**Input:** An directed graph $G = \langle V, E, c \rangle$, where $c : E \to \mathbb{Z}$ is a cost function and a cost value $K$. 
**Definition**

**Input:** An directed graph \( G = (V, E, c) \), where \( c : E \to \mathbb{Z} \) is a cost function and a cost value \( K \).

**Query:** Is there a cycle in \( G \) of cost at least \( K \)?
Longest Circuit

Definition

**Input:** An directed graph $G = (V, E, c)$, where $c : E \rightarrow \mathbb{Z}$ is a cost function and a cost value $K$.

**Query:** Is there a cycle in $G$ of cost at least $K$?

Reduction

NP-completeness

Computational Complexity
Longest Circuit

Definition

Input: An directed graph $G = \langle V, E, c \rangle$, where $c : E \rightarrow \mathbb{Z}$ is a cost function and a cost value $K$.

Query: Is there a cycle in $G$ of cost at least $K$?

Reduction

$\text{DIRECTED-HAMCYCLE} \leq \text{LONGEST-PATH}$. 
Degree-restricted Spanning Tree

Definition

Input: An undirected graph $G = \langle V, E, c \rangle$, where $c: E \rightarrow \mathbb{Z}$ is a cost function, a degree measure $D$ and a cost value $K$.

Query: Is there a spanning tree $T$ of $G$, such that $c(T) \leq K$ and every vertex in $T$ has degree at most $D$?

Reduction

Directed Hamilton Path $\leq$ Directed Spanning Tree.

NP-completeness

Computational Complexity
Degree-restricted Spanning Tree

Definition

Input: An undirected graph $G = \langle V, E, c \rangle$, where $c: E \rightarrow \mathbb{Z}$ is a cost function, a degree measure $D$, and a cost value $K$.
Query: Is there a spanning tree $T$ of $G$, such that $c(T) \leq K$ and every vertex in $T$ has degree at most $D$?

Reduction $D$IRECTED-HAM-PATH $\leq$ DEG-SPANNING-REE.

NP-completeness

Computational Complexity
Degree-restricted Spanning Tree

Definition

**Input:** An undirected graph $G = \langle V, E, c \rangle$, where $c : E \rightarrow \mathbb{Z}$ is a cost function, a degree measure $D$ and a cost value $K$. 
Degree-restricted Spanning Tree

**Definition**

**Input:** An undirected graph $G = \langle V, E, c \rangle$, where $c : E \rightarrow \mathbb{Z}$ is a cost function, a degree measure $D$ and a cost value $K$.

**Query:** Is there a spanning tree $T$ of $G$, such that $c(T) \leq K$ and every vertex in $T$ has degree at most $D$?
Degree-restricted Spanning Tree

Definition

**Input:** An undirected graph $G = \langle V, E, c \rangle$, where $c : E \rightarrow \mathbb{Z}$ is a cost function, a degree measure $D$ and a cost value $K$.

**Query:** Is there a spanning tree $T$ of $G$, such that $c(T) \leq K$ and every vertex in $T$ has degree at most $D$?

Reduction
Degree-restricted Spanning Tree

**Definition**

**Input:** An undirected graph $G = \langle V, E, c \rangle$, where $c : E \rightarrow \mathbb{Z}$ is a cost function, a degree measure $D$ and a cost value $K$.

**Query:** Is there a spanning tree $T$ of $G$, such that $c(T) \leq K$ and every vertex in $T$ has degree at most $D$?

**Reduction**

**DIRECTED-HAMPATH \leq DEG-SPANNING-TREE.**

Computational Complexity
Exact Spanning Tree

**Definition**

- **Input:** An undirected graph $G = \langle V, E, c \rangle$, where $c: E \rightarrow \mathbb{Z}$ is a cost function, and a cost value $K$.

- **Query:** Is there a spanning tree $T$ of $G$, such that $c(T) = K$?

**Reduction** $\text{SUBSET-SUM} \leq \text{EXACT-SPANNING-TREE}$.

**NP-completeness**  Computational Complexity
Exact Spanning Tree

Definition

Input: An undirected graph $G = \langle V, E, c \rangle$, where $c: E \rightarrow \mathbb{Z}$ is a cost function, and a cost value $K$.
Query: Is there a spanning tree $T$ of $G$, such that $c(T) = K$?

Reduction $\text{SUBSET-SUM} \leq \text{EXACT-SUMMING-TREE}$.

NP-completeness
Exact Spanning Tree

Definition

**Input:** An undirected graph $G = \langle V, E, c \rangle$, where $c : E \rightarrow \mathbb{Z}$ is a cost function, and a cost value $K$. 
Exact Spanning Tree

Definition

**Input:** An undirected graph $G = \langle V, E, c \rangle$, where $c : E \to \mathbb{Z}$ is a cost function, and a cost value $K$.

**Query:** Is there a spanning tree $T$ of $G$, such that $c(T) = K$?
Exact Spanning Tree

**Definition**

*Input:* An undirected graph $G = \langle V, E, c \rangle$, where $c : E \rightarrow \mathbb{Z}$ is a cost function, and a cost value $K$.

*Query:* Is there a spanning tree $T$ of $G$, such that $c(T) = K$?

**Reduction**

Reduction $S_{UBSET}$-$SUM \leq EXACT$-$SPANNING$-$TREE$. 

**NP-completeness**

Computational Complexity
Definition

**Input:** An undirected graph $G = \langle V, E, c \rangle$, where $c : E \rightarrow \mathbb{Z}$ is a cost function, and a cost value $K$.

**Query:** Is there a spanning tree $T$ of $G$, such that $c(T) = K$?

Reduction

$\text{SUBSET-SUM} \leq \text{EXACT-SPANNING-TREE}$. 