NP-completeness - Part II

K. Subramani[1]

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

April 6, 2015

Outline

Outline

Outline

Outline

Independent Set

Independent Set

Definition

Independent Set

Definition

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

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 $u, v \in V'$, $(u, v) \notin E$?

Example

Example

Example

Example

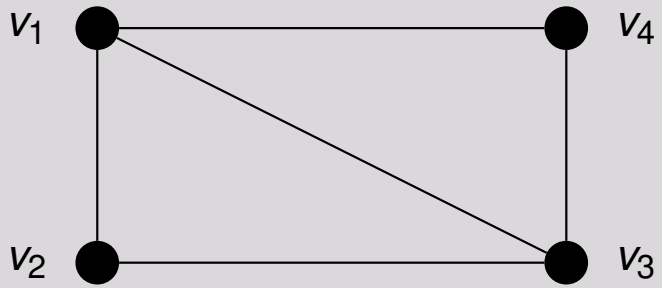
Example

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

Example

Example

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



Complexity

Theorem

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

Complexity

Theorem

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

Proof

Complexity

Theorem

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

Proof

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

Complexity

Theorem

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

Proof

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

Complexity

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 ϕ of 3SAT with m clauses and n variables,

Complexity

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 ϕ of 3SAT with m clauses and n variables, we construct a graph $G = \langle V, E \rangle$ as follows:

Complexity

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 ϕ 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.

Complexity

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 ϕ 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.

Complexity

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 ϕ 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 $v = \neg u$.

Complexity

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 ϕ 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 $K = m$.

Graphical representation

Graphical representation

Example

Graphical representation

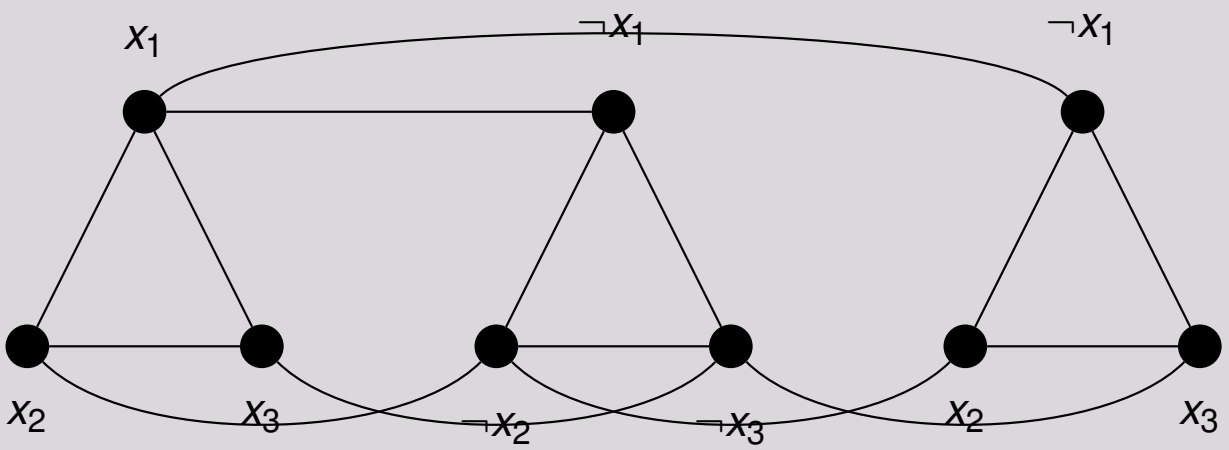
Example

$\phi = (x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3)$

Graphical representation

Example

$\phi = (x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3)$



Completing the Reduction

Completing the Reduction

Proof

Completing the Reduction

Proof

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

Completing the Reduction

Proof

We claim that ϕ 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 ϕ.

Completing the Reduction

Proof

We claim that ϕ 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 ϕ.
2. Pick a node in each clause triangle that is set to **true** under this assignment.

Completing the Reduction

Proof

We claim that ϕ 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 ϕ.
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?

Completing the Reduction

Proof

We claim that ϕ 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 ϕ.
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 ϕ 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 ϕ.
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 $|V'| \geq m$.

Completing the Reduction

Proof

We claim that ϕ 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 ϕ.
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, $|V'| = m$.

Completing the Reduction

Proof

We claim that ϕ 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 ϕ.
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?

Completing the Reduction

Proof

We claim that ϕ 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 ϕ.
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**.

Completing the Reduction

Proof

We claim that ϕ 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 ϕ.
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.

Completing the Reduction

Proof

We claim that ϕ 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 ϕ.
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

Vertex-Cover

Definition

Vertex-Cover

Definition

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

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 $u, v \in V$, $(u, v) \in E$

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 $u, v \in V$, $(u, v) \in E \rightarrow (u \in V')$ or $v \in V'$?

Complexity

Complexity

Theorem

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

Complexity

Theorem

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

Proof

Complexity

Theorem

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

Proof

1. VERTEX-COVER is clearly in **NP**.

Complexity

Theorem

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

Proof

1. VERTEX-COVER is clearly in **NP**.
2. We reduce INDEPENDENT-SET to VERTEX-COVER.

Complexity

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.

Complexity

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 $(G = \langle V, E \rangle, |V| - K)$.

Complexity

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.

Clique

Clique

Definition

Clique

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 $u, v \in V'$, $(u, v) \in E$?

Complexity

Complexity

Theorem

Complexity

Theorem

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

Complexity

Theorem

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

Proof

Complexity

Theorem

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

Proof

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

Complexity

Theorem

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

Proof

1. CLIQUE is clearly in **NP**.
2. We reduce INDEPENDENT-SET to CLIQUE.

Complexity

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.

Complexity

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 $(G^c = \langle V, E^c \rangle, K)$.

Complexity

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

Graph 3-Colorability

Definition

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 $f : V \to C$, such that for all $(u, v) \in E$, $f(u) \neq f(v)$?

Complexity

Complexity

Theorem

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

Complexity

Theorem

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

Proof

Complexity

Theorem

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

Proof

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

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.

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

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 $\phi = C_1 \wedge C_2 \ldots C_m$ 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:

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 $\phi = C_1 \wedge C_2 \ldots C_m$ 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. $V = \{a\}$

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 $\phi = C_1 \wedge C_2 \ldots C_m$ 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. $V = \{a\} \cup \{x_i, \neg x_i\}, \forall i = 1, 2, \ldots, n$

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 $\phi = C_1 \wedge C_2 \ldots C_m$ 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.

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 $\phi = C_1 \wedge C_2 \ldots C_m$ 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. $E_1 = \{a, x_i\}, \forall i = 1, 2, \ldots n$

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 $\phi = C_1 \wedge C_2 \ldots C_m$ 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. $E_1 = \{a, x_i\}, \forall i = 1, 2, \ldots n \cup \{a, \neg x_i\}, \forall i = 1, 2, \ldots n$.

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 $\phi = C_1 \wedge C_2 \ldots C_m$ 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. $E_2 = \{C_{i1}, C_{i2}\}$

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 $\phi = C_1 \wedge C_2 \ldots C_m$ 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. $E_2 = \{C_{i1}, C_{i2}\} \cup \{C_{i1}, C_{i3}\}$

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 $\phi = C_1 \wedge C_2 \ldots C_m$ 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. $E_2 = \{C_{i1}, C_{i2}\} \cup \{C_{i1}, C_{i3}\} \cup \{C_{i2}, C_{i3}\}, \forall i = 1, 2, \ldots, m$.

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 $\phi = C_1 \wedge C_2 \ldots C_m$ 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. $E_3 = \cup\{C_{ij}, x_k\}$,

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 $\phi = C_1 \wedge C_2 \ldots C_m$ 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. $E_3 = \cup\{C_{ij}, x_k\}, \forall j = 1, 2, 3,$

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 $\phi = C_1 \wedge C_2 \ldots C_m$ 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 $C_{ij} = x_k$.

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 $\phi = C_1 \wedge C_2 \ldots C_m$ 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. $E_4 = \cup\{C_{ij}, \neg x_k\}$,

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 $\phi = C_1 \wedge C_2 \ldots C_m$ 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. $E_4 = \cup\{C_{ij}, \neg x_k\}, \forall j = 1, 2, 3,$

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 $\phi = C_1 \wedge C_2 \ldots C_m$ 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. $E_4 = \cup\{C_{ij}, \neg x_k\}, \forall j = 1, 2, 3, \forall i = 1, 2, \ldots, m, \forall k = 1, 2, \ldots n$, if $C_{ij} = \neg x_k$.

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 $\phi = C_1 \wedge C_2 \ldots C_m$ 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. $E_4 = \cup\{C_{ij}, \neg x_k\}, \forall j = 1, 2, 3, \forall i = 1, 2, \ldots, m, \forall k = 1, 2, \ldots n$, if $C_{ij} = \neg x_k$.
   6. $E_5 = \cup\{x_i, \neg x_i\}, \forall i = 1, 2, \ldots n$.

Complexity

Theorem

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

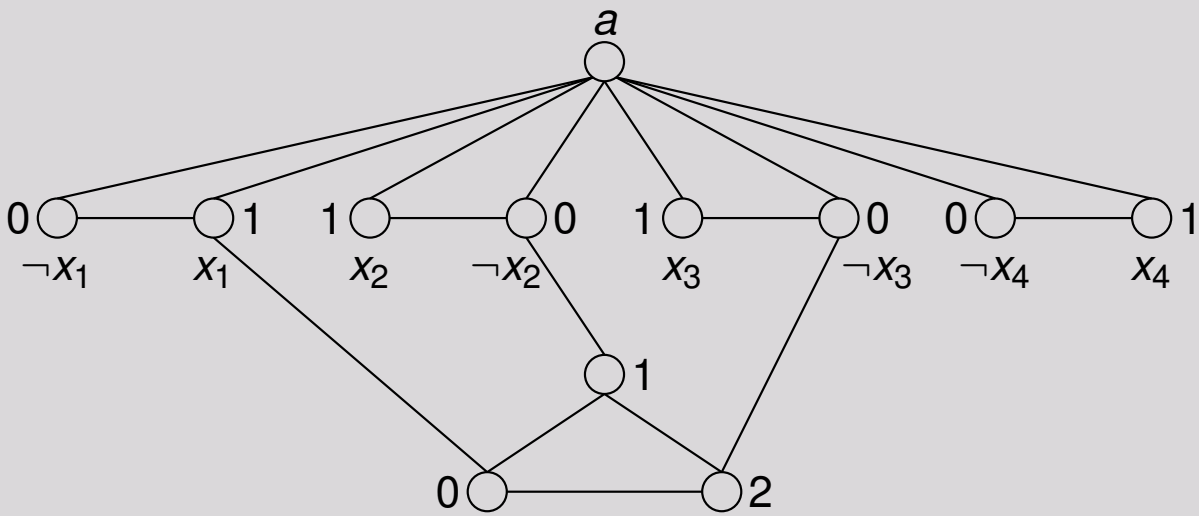
Proof

1. GRAPH 3-COLORABILITY is clearly in **NP**.
2. We reduce NAE3SAT to GRAPH 3-COLORABILITY.
3. Let $\phi = C_1 \wedge C_2 \ldots C_m$ 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. $E_4 = \cup\{C_{ij}, \neg x_k\}, \forall j = 1, 2, 3, \forall i = 1, 2, \ldots, m, \forall k = 1, 2, \ldots n$, if $C_{ij} = \neg x_k$.
   6. $E_5 = \cup\{x_i, \neg x_i\}, \forall i = 1, 2, \ldots n$.
   7. $E = E_1 \cup E_2 \cup E_3 \cup E_4 \cup E_5$.

Example

Example

Example



Construction for ... $\wedge$ $(x_1, \neg x_2, \neg x_3)$ $\wedge$ ...

Argument

Argument

Completing the reduction

Argument

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?)

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,

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**.

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.

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**?

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**?

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 ϕ has a nae-satisfying assignment.

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 ϕ has a nae-satisfying assignment.
7. Color the literals in G as per this assignment and assign color 2 to vertex a.

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 ϕ 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.

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 ϕ 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 ϕ 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

MaxCut

Definition

MaxCut

Definition

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

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

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.

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?

Complexity

Complexity

Theorem

Complexity

Theorem

MAXCUT *is* **NP-complete**.

Complexity

Theorem

MAXCUT *is* **NP-complete**.

Proof

Complexity

Theorem

MAXCUT *is* **NP-complete**.

Proof

1. MAXCUT is clearly in **NP**.

Complexity

Theorem

MaxCut *is* **NP-complete**.

Proof

1. MaxCut is clearly in **NP**.
2. We reduce NAE3SAT to MaxCut.

Complexity

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.

Complexity

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:

Complexity

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. $V = \{x_1, x_2, \ldots x_n\}$

Complexity

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. $V = \{x_1, x_2, \ldots x_n\} \cup \{\neg x_1, \neg x_2, \ldots \neg x_n\}$.

Complexity

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

Complexity

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).

Complexity

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.

Complexity

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. $E = E_1 \cup E_2$.

Complexity

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. $E = E_1 \cup E_2$.
   5. Set $K = 5 \cdot m$.

Example

Example

Example

Let $\phi =$

Example

Example

Let $\phi = (x_1 \vee x_2) \wedge (x_1 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3)$

Example

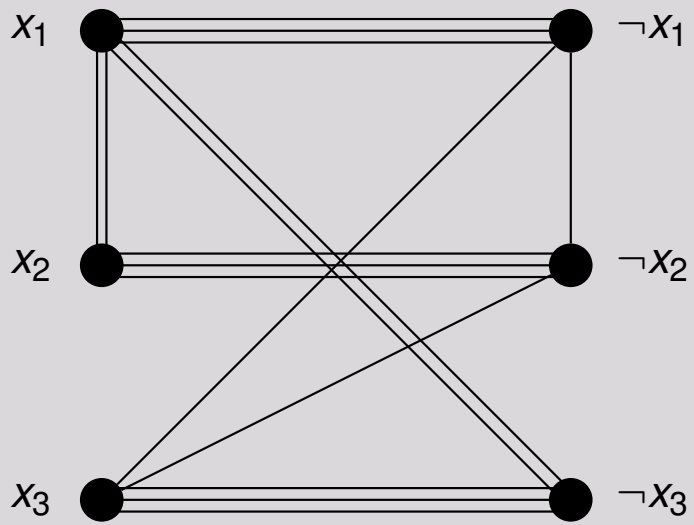
Example

Let $\phi = (x_1 \vee x_2) \wedge (x_1 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3) \equiv$
$(x_1 \vee x_2 \vee x_2) \wedge (x_1 \vee \neg x_3 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3)$

Example

Example

Let $\phi = (x_1 \vee x_2) \wedge (x_1 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3) \equiv$
$(x_1 \vee x_2 \vee x_2) \wedge (x_1 \vee \neg x_3 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3)$



Argument - Part I

Argument - Part I

Lemma

Argument - Part I

Lemma

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

Argument - Part I

Lemma

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

Argument - Part I

Lemma

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

Proof

Argument - Part I

Lemma

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

Argument - Part I

Lemma

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

Argument - Part I

Lemma

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

Argument - Part I

Lemma

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

Argument - Part I

Lemma

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

Argument - Part I

Lemma

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

Argument - Part I

Lemma

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

Argument - Part II

Lemma

Argument - Part II

Lemma

Assume that ϕ has a nae-satisfying assignment.

Argument - Part II

Lemma

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

Argument - Part II

Lemma

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

Proof

Argument - Part II

Lemma

Assume that ϕ 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**.

Argument - Part II

Lemma

Assume that ϕ 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.

Argument - Part II

Lemma

Assume that ϕ 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,

Argument - Part II

Lemma

Assume that ϕ 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.

Argument - Part II

Lemma

Assume that ϕ 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.

Argument - Part II

Lemma

Assume that ϕ 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;

Argument - Part II

Lemma

Assume that ϕ 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

Max-Bisection

Definition

Max-Bisection

Definition

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

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.

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

Example

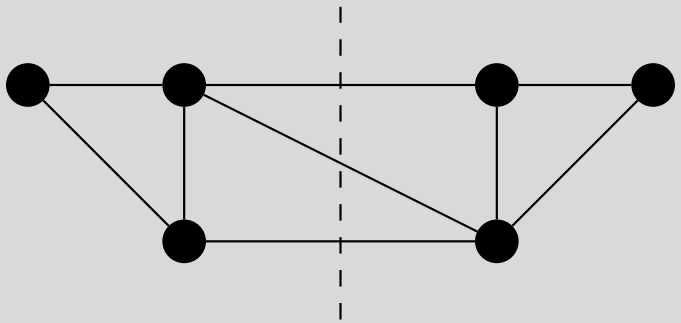
Max-Bisection

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



Complexity

Complexity

Theorem

Complexity

Theorem

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

Complexity

Theorem

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

Proof

Complexity

Theorem

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

Proof

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

Complexity

Theorem

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

Proof

1. MAX-BISECTION is clearly in **NP**.
2. We reduce MAXCUT to MAX-BISECTION.

Complexity

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:

Complexity

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

Complexity

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. $E' = E$.

Complexity

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. $K' = K$.

Argument

Argument

Completing the argument

Argument

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.

Bisection-Width

Bisection-Width

Definition

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 $|S| = |V - S|$?

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.

Bisection-Width

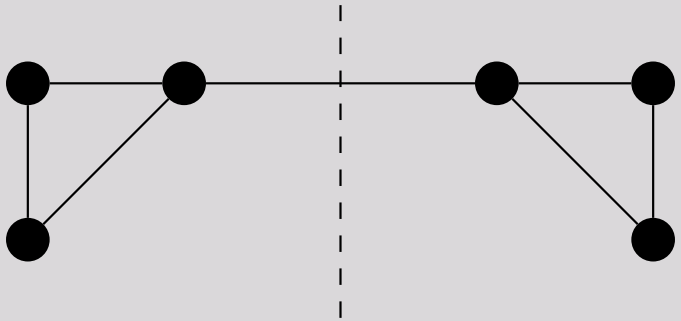
Definition

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



Complexity

Complexity

Theorem

Complexity

Theorem

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

Complexity

Theorem

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

Proof

Complexity

Theorem

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

Proof

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

Complexity

Theorem

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

Proof

1. BISECTION-WIDTH is clearly in **NP**.
2. We reduce MAX-BISECTION to BISECTION-WIDTH.

Complexity

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.

Complexity

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 $|V| = 2 \cdot n$.

Complexity

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?

Complexity

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: $(G^c = \langle V, E^c \rangle, n^2 - K)$.

Complexity

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

Subset-Sum

Definition

Subset-Sum

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 $\sum_{a_i \in S'} a_i = T$?

Complexity

Complexity

Theorem

Complexity

Theorem

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

Complexity

Theorem

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

Proof

Complexity

Theorem

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

Proof

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

Complexity

Theorem

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

Proof

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

Complexity

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 $\phi = C_1 \wedge C_2 \wedge \ldots \wedge C_m$ of m clauses over n variables, we construct the following instance of SUBSET-SUM:

Complexity

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 $\phi = C_1 \wedge C_2 \wedge \ldots \wedge C_m$ 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.

Complexity

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 $\phi = C_1 \wedge C_2 \wedge \ldots \wedge C_m$ 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.

Complexity

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 $\phi = C_1 \wedge C_2 \wedge \ldots \wedge C_m$ 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 Sl_1 and Sl_2.

Complexity

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 $\phi = C_1 \wedge C_2 \wedge \ldots \wedge C_m$ 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 Sl_1 and Sl_2.
    4. Finally, we create a target which has 1 in the first n digits and 4 in the final m digits.

Example

Example

Example

Example

Example

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

Example

Example

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

The corresponding instance of SUBSET-SUM is given below:

Example

Example

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

The corresponding instance of SUBSET-SUM is given below:

	x_1	x_2	x_3	x_4	c_1	c_2
T_1	1	0	0	0	1	0
F_1	1	0	0	0	0	1
T_2	0	1	0	0	0	1
F_2	0	1	0	0	0	0
T_3	0	0	1	0	0	0
F_3	0	0	1	0	1	0
T_4	0	0	0	1	0	0
F_4	0	0	0	1	1	1
$S1_1$	0	0	0	0	1	0
$S1_2$	0	0	0	0	2	0
$S2_1$	0	0	0	0	0	1
$S2_2$	0	0	0	0	0	2
Target	1	1	1	1	4	4

Argument

Argument

Proof

Argument

Proof

1. Assume that ϕ is satisfiable.

Argument

Proof

1. Assume that ϕ is satisfiable.
2. Pick all the rows that correspond to **true** literals.

Argument

Proof

1. Assume that ϕ 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.

Argument

Proof

1. Assume that ϕ 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

Proof

1. Assume that ϕ 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.

Argument

Proof

1. Assume that ϕ 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 Sl_1 and Sl_2, or Sl_2 or Sl_1 respectively.
6. Clearly the final m bits of the target are met.

Argument

Proof

1. Assume that ϕ 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 Sl_1 and Sl_2, or Sl_2 or Sl_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.

Argument

Proof

1. Assume that ϕ 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 Sl_1 and Sl_2, or Sl_2 or Sl_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 ϕ 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 Sl_1 and Sl_2, or Sl_2 or Sl_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 ϕ 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 Sl_1 and Sl_2, or Sl_2 or Sl_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.

Argument

Proof

1. Assume that ϕ 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 Sl_1 and Sl_2, or Sl_2 or Sl_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

Partition

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 $\sum_{a_j \in S'} a_j = \sum_{a_j \in S - S'} a_j$?

Complexity

Complexity

Theorem

Complexity

Theorem

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

Complexity

Theorem

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

Proof

Complexity

Theorem

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

Proof

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

Complexity

Theorem

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

Proof

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

Complexity

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.

Complexity

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:

Complexity

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 $L = \sum_{a_i \in S} a_i$.

Argument

Argument

Completing the argument

Argument

Completing the argument

1. Assume that S has a subset S' which sums 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 $S \setminus S' \cup \{L + T\}$.

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

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 $2 \cdot L$!

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).

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

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 $2 \cdot L$.

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?

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

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

Knapsack

Definition

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.

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.

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

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.

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

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.

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

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

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.

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

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

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

Complexity

Complexity

Theorem

Complexity

Theorem

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

Complexity

Theorem

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

Proof

Complexity

Theorem

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

Proof

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

Complexity

Theorem

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

Proof

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

Complexity

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:

Complexity

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 $w_i = p_i = a_i, \forall i = 1, 2, \ldots n$.

Complexity

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

Complexity

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 $w_i = p_i = a_i, \forall i = 1, 2, \ldots n$.
   2. Set $W = P = T$.
   3. Can you establish that the instance of SUBSET-SUM is **true** if and only if the instance of KNAPSACK is?

The Power of Integer Programming

The Power of Integer Programming

Exercise

Reduce all the problems discussed thus far to Integer Programming.

Directed Hamilton Path

Directed Hamilton Path

Definition

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

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.

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.

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

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

3SAT $\leq$ DIRECTED-HAMPATH.

$s - t$ Directed Hamilton Path

$s - t$ Directed Hamilton Path

Definition

$s - t$ Directed Hamilton Path

Definition

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

$s - t$ 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 $G = \langle V, E \rangle$ 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.

$s - t$ 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?

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

Reduction

$s - t$ 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?

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

Reduction

Same as above.

Directed Hamilton Circuit

Directed Hamilton Circuit

Definition

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

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?

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.

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

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

$s - t$ DIRECTED-HAMPATH $\leq$ DIRECTED-HAMCYCLE.

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

$s - t$ DIRECTED-HAMPATH $\leq$ 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

$s - t$ DIRECTED-HAMPATH $\leq$ DIRECTED-HAMCYCLE.

Exercise

Can you provide a reduction from DIRECTED-HAMPATH *to* DIRECTED-HAMCYCLE*?*

Undirected Hamilton 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 $G = \langle V, E \rangle$.

Query: Is there an undirected Hamilton cycle in G?

Reduction

Undirected Hamilton Cycle

Definition

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

Query: Is there an undirected Hamilton cycle in G?

Reduction

DIRECTED-HAMCYCLE $\leq$ HAMCYCLE.

Traveling Salesman Problem

Traveling Salesman Problem

Definition

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

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

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

DIRECTED-HAMCYCLE $\leq$ TSP(D).

Traveling Salesman Problem (Triangle Inequality)

Traveling Salesman Problem (Triangle Inequality)

Definition

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

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):

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):

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

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?

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

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

DIRECTED-HAMCYCLE $\leq \triangle$TSP(D).

Longest Path

Longest Path

Definition

Longest Path

Definition

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

Longest Path

Definition

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

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

Longest Path

Definition

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

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

Reduction

DIRECTED-HAMPATH $\leq$ LONGEST-PATH.

Longest Circuit

Longest Circuit

Definition

Longest Circuit

Definition

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

Longest Circuit

Definition

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

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

Reduction

Longest Circuit

Definition

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

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

Reduction

DIRECTED-HAMCYCLE $\leq$ LONGEST-PATH.

Degree-restricted Spanning Tree

Degree-restricted Spanning Tree

Definition

Degree-restricted Spanning Tree

Definition

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

Degree-restricted Spanning Tree

Definition

Input: An undirected graph $G = \langle V, E, \mathbf{c} \rangle$, where $\mathbf{c} : E \to \mathcal{Z}$ 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 $G = \langle V, E, \mathbf{c} \rangle$, where $\mathbf{c} : E \to \mathcal{Z}$ 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 $G = \langle V, E, \mathbf{c} \rangle$, where $\mathbf{c} : E \to \mathcal{Z}$ 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.

Exact Spanning Tree

Exact Spanning Tree

Definition

Exact Spanning Tree

Definition

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

Exact Spanning Tree

Definition

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

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

Reduction

Exact Spanning Tree

Definition

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

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

Reduction

SUBSET-SUM $\leq$ EXACT-SPANNING-TREE.