Chapter 5
$\mathcal{NP}$ and Computational Intractability

Advanced algorithms on February 25, 2016

Huynh Tuong Nguyen
Faculty of Computer Science and Engineering
University of Technology - VNUHCM

Acknowledgement

The material in this slide based on

- Stephen Cook (1971). *The Complexity of Theorem Proving Procedures*. Proceedings of the third annual ACM symposium on Theory of computing. pp. 151–158.
- Richard M. Karp (1972). *Reducibility Among Combinatorial Problems*. In R. E. Miller and J. W. Thatcher (editors). Complexity of Computer Computations. New York: Plenum. pp. 85–103.
- Michael R. Garey, David S. Johnson (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness* (Series of Books in the Mathematical Sciences). San Francisco, Calif.: W. H. Freeman.

Contents

Evolution of computer-based resolution

Resolution methods

- Pb. A
- Pb. B
- Pb. C
- Pb. D

Pb.: *problem*
Algo.: *algorithm*

Evolution of computer-based resolution

Resolution methods

- Pb. A $\leftarrow$ Algo. Π_A
- Pb. B $\leftarrow$ Algo. Π_B
- Pb. C $\leftarrow$ Algo. Π_C
- Pb. D $\leftarrow$ Algo. Π_D

Pb.: *problem*
Algo.: *algorithm*

Evolution of computer-based resolution

Resolution methods

- Pb. $A \leftarrow$ Algo. Π_A
- Pb. $B \leftarrow$ Algo. Π_B
- Pb. $C \leftarrow$ Algo. Π_C
- Pb. $D \leftarrow$ Algo. Π_D

Pb.: *problem*
Algo.: *algorithm*

Difficulty relation between Pbs.

- Pb. $A \leftarrow$ Algo. Π_A
- If Pb. $B \preceq$ Pb. A

$\preceq$: *is more easy to solve than*
$\Pi_{A'}$: some simple modification of Π_A

Evolution of computer-based resolution

Resolution methods

- Pb. $A \leftarrow$ Algo. Π_A
- Pb. $B \leftarrow$ Algo. Π_B
- Pb. $C \leftarrow$ Algo. Π_C
- Pb. $D \leftarrow$ Algo. Π_D

Pb.: *problem*
Algo.: *algorithm*

Difficulty relation between Pbs.

- Pb. $A \leftarrow$ Algo. Π_A
- If Pb. $B \preceq$ Pb. A
- $\exists\ \Pi_{A'}$: Pb. $B \leftarrow$ Algo. $\Pi_{A'}$ (i.e. $\Pi_{A'} \equiv \Pi_B$)

$\preceq$: *is more easy to solve than*
$\Pi_{A'}$: some simple modification of Π_A

Evolution of computer-based resolution

Resolution methods

- Pb. $A \leftarrow$ Algo. Π_A
- Pb. $B \leftarrow$ Algo. Π_B
- Pb. $C \leftarrow$ Algo. Π_C
- Pb. $D \leftarrow$ Algo. Π_D

Pb.: *problem*
Algo.: *algorithm*

Difficulty relation between Pbs.

- Pb. $A \leftarrow$ Algo. Π_A
- If Pb. $B \preceq$ Pb. A
- $\exists\ \Pi_{A'}$: Pb. $B \leftarrow$ Algo. $\Pi_{A'}$ (i.e. $\Pi_{A'} \equiv \Pi_B$)

$\preceq$: *is more easy to solve than*
$\Pi_{A'}$: some simple modification of Π_A

$\exists$ Pb. X, $\exists$ Algo. Π_X: $(A, B, C, D \preceq X) \wedge (X \leftarrow \Pi_X)$?

Exercise

Problem HAMILTONIAN-CYCLE

Input: Given a digraph $G = (V, E)$.
Goal: Decide whether there exists simple cycle that contains every node in V.

Exercise

Problem HAMILTONIAN-CYCLE

Input: Given a digraph $G = (V, E)$.
Goal: Decide whether there exists simple cycle that contains every node in V.

Problem LONGEST-SIMPLE-PATH

Input: Given digraph $G = (V, E)$ where $V = \{v_1, v_2, ..., v_n\}$ be a set of nodes and $d(v_i, v_j)$ the distance between v_i and v_j.
Goal: Find a longest simple path from node u to node v.

Solution

Algorithm

1. Set the length of each edge in G to be 1
2. for each edge $(u, v) \in E$ do
3. find the longest simple path P from u to v in G.
4. if the length of P is $n - 1$ then
5. by adding edge (u, v) we obtain an Hamilton circuit in G.
6. if no Hamilton circuit is found for every (u, v) then
7. print "no Hamilton circuit exists"

Solution

Algorithm

1. Set the length of each edge in G to be 1
2. for each edge $(u, v) \in E$ do
3. find the longest simple path P from u to v in G.
4. if the length of P is $n - 1$ then
5. by adding edge (u, v) we obtain an Hamilton circuit in G.
6. if no Hamilton circuit is found for every (u, v) then
7. print "no Hamilton circuit exists"

If LONGEST-SIMPLE-PATH can be solved in polynomial time, then HAMILTONIAN-CYCLE can also be solved in polynomial time.

Solution

Algorithm

1. Set the length of each edge in G to be 1
2. for each edge $(u, v) \in E$ do
3. find the longest simple path P from u to v in G.
4. if the length of P is $n - 1$ then
5. by adding edge (u, v) we obtain an Hamilton circuit in G.
6. if no Hamilton circuit is found for every (u, v) then
7. print "no Hamilton circuit exists"

If LONGEST-SIMPLE-PATH can be solved in polynomial time, then HAMILTONIAN-CYCLE can also be solved in polynomial time.

Since HAMILTONIAN-CYCLE was proved to be $\mathcal{NP}$-complete, LONGEST-SIMPLE-PATH is also $\mathcal{NP}$-complete.

Decision problem

Definition

A 'y/n' question, with input of size n.

Decision problem

Definition

A 'y/n' question, with input of size n.

Example

- Given k and n values, is k the maximum one among the values?

Optimization problem

Definition

Finding the best answer to a particular problem.

Optimization problem

Definition

Finding the best answer to a particular problem.

Example

- Given a set $\mathcal{S}$ of integer values and a fitness function $f : \mathcal{S} \mapsto \mathcal{R}$, find a value $k \in \mathcal{S}$ such that $f(k)$ is minimized?

Polynomial time

An algorithm A runs in polynomial time if for every input x, A terminates in at most $p(|x|)$ "steps", where $p(.)$ is some polynomial and $|x|$ is length of x.

Polynomial time

An algorithm A runs in polynomial time if for every input x, A terminates in at most $p(|x|)$ "steps", where $p(.)$ is some polynomial and $|x|$ is length of x.

Certifier

An algorithm to check that a solution answers ("yes" or "no" for decision problem).

$\mathcal{NP}$

Polynomial time

An algorithm A runs in polynomial time if for every input x, A terminates in at most $p(|x|)$ "steps", where $p(.)$ is some polynomial and $|x|$ is length of x.

Certifier

An algorithm to check that a solution answers ("yes" or "no" for decision problem).

$\mathcal{NP}$

Decision problems for which there is a polynomial-time certifier.

Certifiers and Certificates: Hamiltonian Cycle

Problem HAMILTONIAN-CYCLE

Input: Given a digraph $G = (V, E)$.
Goal: Does there exists simple cycle that visites every node?

Certifiers and Certificates: Hamiltonian Cycle

Problem HAMILTONIAN-CYCLE

Input: Given a digraph $G = (V, E)$.
Goal: Does there exists simple cycle that visites every node?

Certificate

A permutation of the n nodes.

Certifiers and Certificates: Hamiltonian Cycle

Problem HAMILTONIAN-CYCLE

Input: Given a digraph $G = (V, E)$.
Goal: Does there exists simple cycle that visites every node?

Certificate

A permutation of the n nodes.

Certifier

Check that the permutation contains each node in V exactly once, and that there is an edge between each pair of adjacent nodes in the permutation.

Certifiers and Certificates: Hamiltonian Cycle

Problem HAMILTONIAN-CYCLE

Input: Given a digraph $G = (V, E)$.
Goal: Does there exists simple cycle that visites every node?

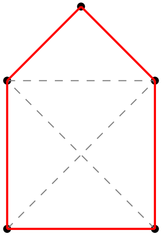
Certificate

A permutation of the n nodes.

Certifier

Check that the permutation contains each node in V exactly once, and that there is an edge between each pair of adjacent nodes in the permutation.

HAM-CYCLE is in $\mathcal{NP}$.



Certifiers and Certificates: Hamiltonian Cycle

Problem HAMILTONIAN-CYCLE

Input: Given a digraph $G = (V, E)$.
Goal: Does there exists simple cycle that visites every node?

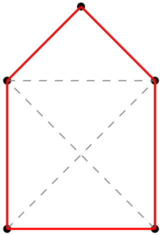
Certificate

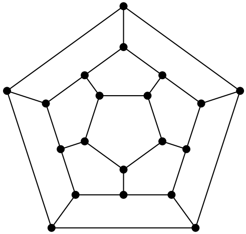
A permutation of the n nodes.

Certifier

Check that the permutation contains each node in V exactly once, and that there is an edge between each pair of adjacent nodes in the permutation.

HAM-CYCLE is in $\mathcal{NP}$.





Certifiers and Certificates: Hamiltonian Cycle

Problem HAMILTONIAN-CYCLE

Input: Given a digraph $G = (V, E)$.
Goal: Does there exists simple cycle that visites every node?

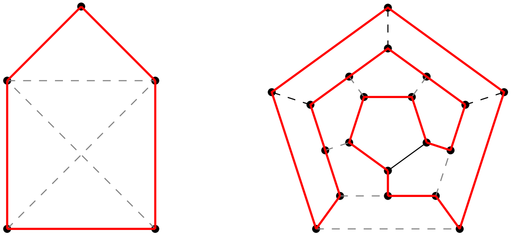
Certificate

A permutation of the n nodes.

Certifier

Check that the permutation contains each node in V exactly once, and that there is an edge between each pair of adjacent nodes in the permutation.

HAM-CYCLE is in $\mathcal{NP}$.



Certifiers and Certificates: Hamiltonian Cycle

Problem HAMILTONIAN-CYCLE

Input: Given a digraph $G = (V, E)$.
Goal: Does there exists simple cycle that visites every node?

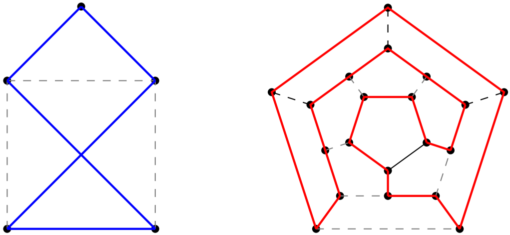
Certificate

A permutation of the n nodes.

Certifier

Check that the permutation contains each node in V exactly once, and that there is an edge between each pair of adjacent nodes in the permutation.

HAM-CYCLE is in $\mathcal{NP}$.



Certifiers and Certificates: Partition

Problem PARTITION

Input: Given a set $\mathcal{A}$ of n integer positive values $\{a_1, a_2, \ldots, a_n\}$ and $\sum_{i=1}^{n} a_i = 2B$.
Goal: Decide whether there is a split the integers into two subsets which sum to half ($= B$).

Certificate

A subset $\mathcal{A}_1$ of $\mathcal{A}$.

Certifier

Check whether $\sum_{a_i \in \mathcal{A}_1} a_i = B$.

PARTITION is in $\mathcal{NP}$.

$\mathcal{P}$, $\mathcal{NP}$, $\mathcal{EXP}$

$\mathcal{P}$: Decision problems for which there is a polynomial-time algorithm.
$\mathcal{EXP}$: Decision problems for which there is an exponential-time algorithm.
$\mathcal{NP}$: Decision problems for which there is a polynomial-time certifier.

$\mathcal{P}$, $\mathcal{NP}$, $\mathcal{EXP}$

$\mathcal{P}$: Decision problems for which there is a polynomial-time algorithm.
$\mathcal{EXP}$: Decision problems for which there is an exponential-time algorithm.
$\mathcal{NP}$: Decision problems for which there is a polynomial-time certifier.

$\mathcal{P} \subseteq \mathcal{NP}$

Consider any decision problem X in $\mathcal{P}$.
$\Rightarrow$ there exists a polynomial-time algorithm A that solves X.
It means that with an input, we know that it answers in polynomial-time (i.e., $\exists$ polynomial-time certifier).

$\mathcal{P}$, $\mathcal{NP}$, $\mathcal{EXP}$

$\mathcal{P}$: Decision problems for which there is a polynomial-time algorithm.
$\mathcal{EXP}$: Decision problems for which there is an exponential-time algorithm.
$\mathcal{NP}$: Decision problems for which there is a polynomial-time certifier.

$\mathcal{P} \subseteq \mathcal{NP}$

Consider any decision problem X in $\mathcal{P}$.
$\Rightarrow$ there exists a polynomial-time algorithm A that solves X.
It means that with an input, we know that it answers in polynomial-time (i.e., $\exists$ polynomial-time certifier).

$\mathcal{NP} \subseteq \mathcal{EXP}$

Consider any problem X in $\mathcal{NP}$.
By definition, there exists a polynomial-time certifier C for X.
To solve input x, run C on all possible certificates (exponential quantity).
Return 'yes', if C returns 'yes' for any of these.

Polynomial-time reduction

Main idea

We can solve A in polynomial time if we can solve B in polynomial time!

Reduction

Problem A polynomial-time reduces to problem B if arbitrary instances of problem A can be solved using:

- Polynomial number of standard computational steps, plus
- Polynomial number of calls to oracle that solves problem B.

Notation

$A \leq_p B$

Remarks

Pay for time to write down instances sent to black box $\Rightarrow$ instances of B must be of polynomial size.
Note: Cook reducibility (vs. Karp reducibility)

Karp and Cook reductions [1971]

Karp reductions

Problem A is polynomial-time (Karp) reducible to Problem B if any instance of problem A can be solved using

- Polynomially many standard computation steps, plus
- One call on some instance to an (black-box) algorithm that solves Problem B

Cook reductions

Problem A is polynomial-time (Cook) reducible to Problem B if any instance of problem A can be solved using

- Polynomially many standard computation steps, plus
- Polynomially many calls on some instance(s) to an (black-box) algorithm that solves Problem B

Contents

Polynomial-time reduction

Purpose

Classify problems according to relative difficulty.

Design algorithms

If $A \leq_p B$ and B can be solved in polynomial-time, then A can be solved in polynomial-time.

Etablish intractability

If $A \leq_p B$ and A cannot be solved in polynomial-time, then B cannot be solved in polynomial-time.

Etablish equivalence

If $A \leq_p B$ and $B \leq_p A$, then $A \cong_p B$.

Vertex Cover

Problem VERTEX-COVER (Phủ đỉnh)

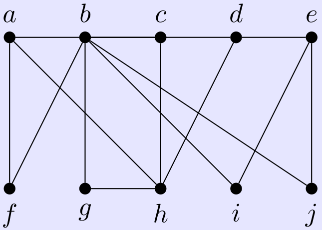
Input: Given a graph $G = (V, E)$ and an integer k.
Goal: Decide whether there is a vertex cover of size $\leq k$.
Vertex cover: vertice subset S ($S \subseteq V$) such that for each edge in G, at least one of its endpoints is in S.

Vertex Cover

Problem VERTEX-COVER (Phủ đỉnh)

Input: Given a graph $G = (V, E)$ and an integer k.
Goal: Decide whether there is a vertex cover of size $\leq k$.
Vertex cover: vertice subset S ($S \subseteq V$) such that for each edge in G, at least one of its endpoints is in S.

Example

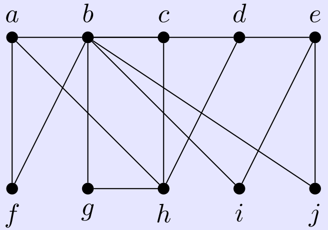


Vertex Cover

Problem VERTEX-COVER (Phủ đỉnh)

Input: Given a graph $G = (V, E)$ and an integer k.
Goal: Decide whether there is a vertex cover of size $\leq k$.
Vertex cover: vertice subset S ($S \subseteq V$) such that for each edge in G, at least one of its endpoints is in S.

Example



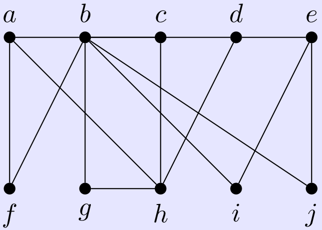
- Is there a vertex cover of size ≤ 6?
- Is there a vertex cover of size ≤ 3?

Vertex Cover

Problem VERTEX-COVER (Phủ đỉnh)

Input: Given a graph $G = (V, E)$ and an integer k.
Goal: Decide whether there is a vertex cover of size $\leq k$.
Vertex cover: vertice subset S ($S \subseteq V$) such that for each edge in G, at least one of its endpoints is in S.

Example



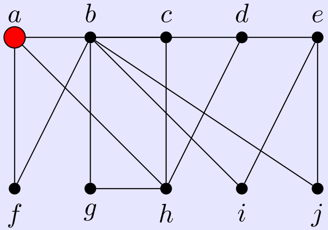
- Is there a vertex cover of size ≤ 6? Yes.
- Is there a vertex cover of size ≤ 3?

Vertex Cover

Problem VERTEX-COVER (Phủ đỉnh)

Input: Given a graph $G = (V, E)$ and an integer k.
Goal: Decide whether there is a vertex cover of size $\leq k$.
Vertex cover: vertice subset S ($S \subseteq V$) such that for each edge in G, at least one of its endpoints is in S.

Example



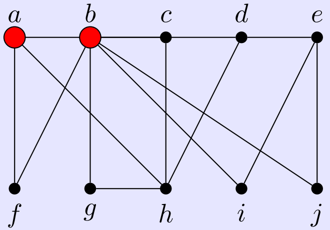
- Is there a vertex cover of size ≤ 6? Yes.
- Is there a vertex cover of size ≤ 3? No.

Vertex Cover

Problem VERTEX-COVER (Phủ đỉnh)

Input: Given a graph $G = (V, E)$ and an integer k.
Goal: Decide whether there is a vertex cover of size $\leq k$.
Vertex cover: vertice subset S ($S \subseteq V$) such that for each edge in G, at least one of its endpoints is in S.

Example



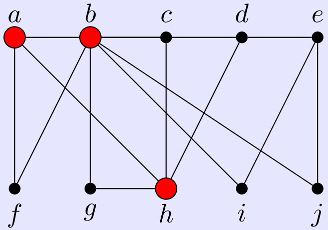
- Is there a vertex cover of size ≤ 6? Yes.
- Is there a vertex cover of size ≤ 3? No.

Vertex Cover

Problem VERTEX-COVER (Phủ đỉnh)

Input: Given a graph $G = (V, E)$ and an integer k.
Goal: Decide whether there is a vertex cover of size $\leq k$.
Vertex cover: vertice subset S ($S \subseteq V$) such that for each edge in G, at least one of its endpoints is in S.

Example



- Is there a vertex cover of size ≤ 6? Yes.
- Is there a vertex cover of size ≤ 3? No.

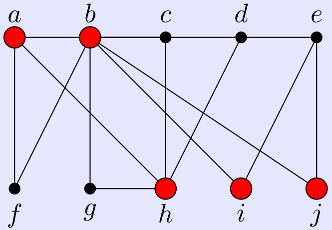
Contents

Vertex Cover

Problem VERTEX-COVER (Phủ đỉnh)

Input: Given a graph $G = (V, E)$ and an integer k.
Goal: Decide whether there is a vertex cover of size $\leq k$.
Vertex cover: vertice subset S ($S \subseteq V$) such that for each edge in G, at least one of its endpoints is in S.

Example



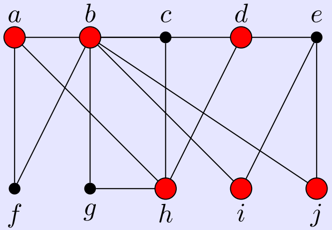
- Is there a vertex cover of size ≤ 6? Yes.
- Is there a vertex cover of size ≤ 3? No.

Vertex Cover

Problem VERTEX-COVER (Phủ đỉnh)

Input: Given a graph $G = (V, E)$ and an integer k.
Goal: Decide whether there is a vertex cover of size $\leq k$.
Vertex cover: vertice subset S ($S \subseteq V$) such that for each edge in G, at least one of its endpoints is in S.

Example



- Is there a vertex cover of size ≤ 6? Yes.
- Is there a vertex cover of size ≤ 3? No.

Independent set

Problem INDEPEDENT-SET (tập đỉnh độc lập)

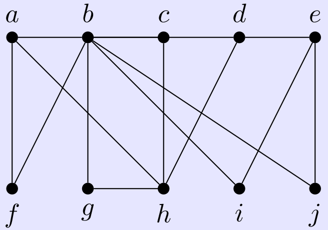
Input: Given a graph $G = (V, E)$ and an integer k.
Goal: Decide whether there is an independent set of size $\geq k$.
Independent set: a set S of size $\geq k$ such that no two nodes in S are connected by an edge of E.

Independent set

Problem INDEPEDENT-SET (tập đỉnh độc lập)

Input: Given a graph $G = (V, E)$ and an integer k.
Goal: Decide whether there is an independent set of size $\geq k$.
Independent set: a set S of size $\geq k$ such that no two nodes in S are connected by an edge of E.

Example

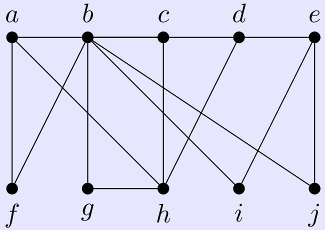


Independent set

Problem INDEPEDENT-SET (tập đỉnh độc lập)

Input: Given a graph $G = (V, E)$ and an integer k.
Goal: Decide whether there is an independent set of size $\geq k$.
Independent set: a set S of size $\geq k$ such that no two nodes in S are connected by an edge of E.

Example



- Is there a vertex cover of size ≥ 5?
- Is there a vertex cover of size ≥ 6?

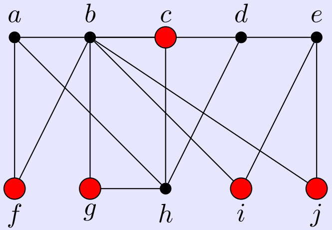
Contents

Independent set

Problem INDEPEDENT-SET (tập đỉnh độc lập)

Input: Given a graph $G = (V, E)$ and an integer k.
Goal: Decide whether there is an independent set of size $\geq k$.
Independent set: a set S of size $\geq k$ such that no two nodes in S are connected by an edge of E.

Example



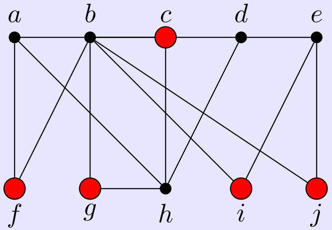
- Is there a vertex cover of size ≥ 5? Yes.
- Is there a vertex cover of size ≥ 6?

Independent set

Problem INDEPEDENT-SET (tập đỉnh độc lập)

Input: Given a graph $G = (V, E)$ and an integer k.
Goal: Decide whether there is an independent set of size $\geq k$.
Independent set: a set S of size $\geq k$ such that no two nodes in S are connected by an edge of E.

Example



- Is there a vertex cover of size ≥ 5? Yes.
- Is there a vertex cover of size ≥ 6? No.

Set Cover

Problem SET-COVER (phủ tập)

Input: Given a set U of elements, a collection $S_1, S_2, \ldots, S_m$ of subsets of U, and an integer k.
Goal: Decide whether there exists a collection of $\leq k$ of these sets whose union is equal to U.

Example

- m available pieces of software.
- Set U of n capabilities that we would like our system to have.
- The i^{th} piece of software provides the set $S_i \subset U$ of capabilities.
- Goal: achieve all n capabilities using fewest pieces of software.

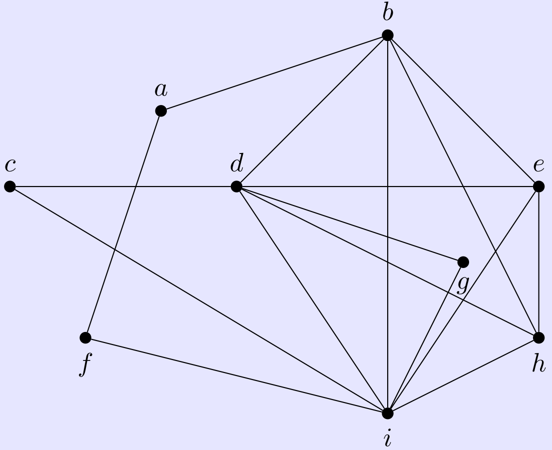
$U = \{1, 2, 3, 4, 5, 6, 7\}$; $k = 2$.
$\Rightarrow S_1 = \{3, 7\}; S_2 = \{3, 4, 5, 6\}; S_3 = \{1\};$
$S_4 = \{2, 4\}; S_5 = \{5\}; S_6 = \{1, 2, 6, 7\};$

Clique

Problem CLIQUE

Input: Given a graph $G = (V, E)$ and a threshold k.
Goal: Decide whether there exists a clique of size k.
Clique: a set $C = \{v_1, v_2, \ldots, v_k\} \subset V$ satisfies $(u, v) \in E$, $\forall u, v \in C$.

Example

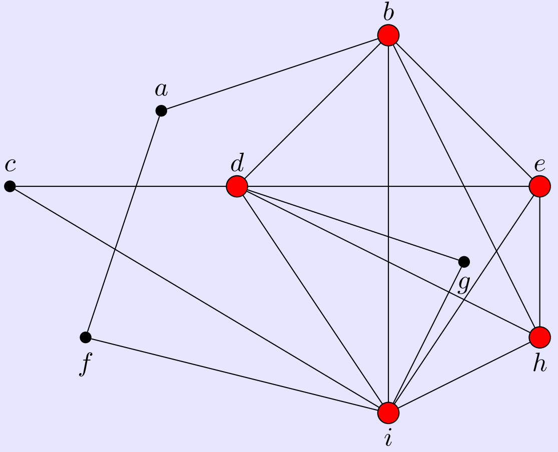


Clique

Problem CLIQUE

Input: Given a graph $G = (V, E)$ and a threshold k.
Goal: Decide whether there exists a clique of size k.
Clique: a set $C = \{v_1, v_2, \ldots, v_k\} \subset V$ satisfies $(u, v) \in E$, $\forall u, v \in C$.

Example



Satisfiability

Problem SAT

Input: Given conjunctive normal form with n variables, x_1, x_2, ..., x_n.
Goal: Decide whether there is an assignment of x_1, x_2, ..., x_n (setting each x_i to be 0 or 1) such that the formula is true (satisfied).

Contents

Satisfiability

Problem SAT

Input: Given conjunctive normal form with n variables, x_1, x_2, ..., x_n.
Goal: Decide whether there is an assignment of x_1, x_2, ..., x_n (setting each x_i to be 0 or 1) such that the formula is true (satisfied).

Problem 3-SAT

A sub problem of SAT where each clause contains exactly 3 literals.

Path and Cycle

Problem HAMILTONIAN-PATH

Input: Given a digraph $G = (V, E)$.
Goal: Decide whether there exists simple path that contains every node in V.

Problem HAMILTONIAN-CYCLE

Input: Given a digraph $G = (V, E)$.
Goal: Decide whether there exists simple cycle that contains every node in V.

Path and Cycle

Problem HAMILTONIAN-PATH

Input: Given a digraph $G = (V, E)$.
Goal: Decide whether there exists simple path that contains every node in V.

Problem HAMILTONIAN-CYCLE

Input: Given a digraph $G = (V, E)$.
Goal: Decide whether there exists simple cycle that contains every node in V.

Problem LONGEST-PATH

Input: Given digraph $G = (V, E)$ and a positive integer k.
Goal: Decide whether there exists a simple path of length at least k edges.

Problem TSP

Input: Given a set of n cities and a pairwise distance function $d(u, v)$.
Goal: Decide whether there is a tour of length $\leq D$.

Numerical Problems

Problem PARTITION

Input: Given a set of n integer positive values $\{a_1, a_2, \ldots, a_n\}$ and $\sum_{i=1}^{n} a_i = 2B$.
Goal: Decide whether there is a split the integers into two subsets which sum to half ($= B$).

Problem SUBSET-SUM

Input: Given an integer positive value m and a set of n integer positive values $\{a_1, a_2, \ldots, a_n\}$.
Goal: Decide whether there exists a subset which sums to m.

Basic reduction strategies

Basic reduction strategies

- Reduction by simple equivalence
- Reduction from special case to general case
- Reduction by encoding with gadgets

Reduction by simple equivalence

VERTEX-COVER $\cong_p$ INDEPENENT-SET

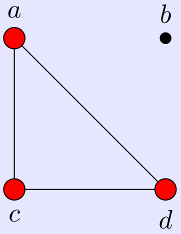
S is an independent set *iff* $V - S$ is a vertex cover.

Reduction by simple equivalence

VERTEX-COVER $\cong_p$ INDEPENENT-SET

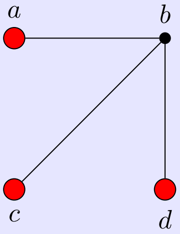
S is an independent set *iff* $V - S$ is a vertex cover.

INDEPENENT-SET $\cong_p$ CLIQUE



IFF

$\Leftrightarrow$



A clique of size k in a graph

An independent set of size k in its complement.

Reduction from special case to general case

3-SAT $\leq_p$ SAT

Trivial

PARTITION $\leq_p$ SUBSET-SUM

Since PARTITION is a special case of SUBSET-SUM.

HAMILTONIAN-PATH $\leq_p$ HAMILTONIAN-CYCLE

Given an instance $G = (V, E)$ of HAMILTONIAN-PATH, create an instance of HAMILTONIAN-CYCLE by adding a dummy node which connect to all vertex $v \in V$.

Reduction from special case to general case

HAMILTONIAN-CYCLE $\leq_p$ TSP

Given an instance $G = (V, E)$ of HAMILTONIAN-CYCLE, create n cities with distance function.
$d(u, v) = 1$ if $(u, v) \in E$, and $= 2$ if $(u, v) \notin E$
TSP instance has tour of length $\leq n$ iif G is Hamiltonian.

Reduction by encoding with gadgets

SUBSET-SUM $\leq_p$ PARTITION

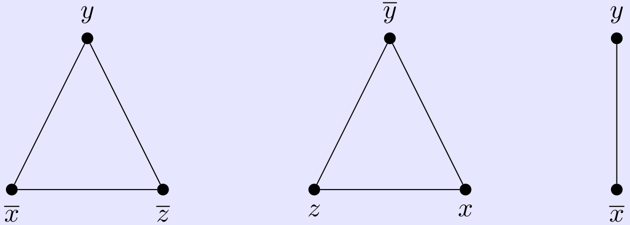
- Given a SUBSET-SUM instance, we construct a PARTITION instance (the construction given by a polynomial transformation): 'adding a new element x in the set with $x = |\sum_{i=1}^{n} a_i - 2m|$'.
- Then, show that SUBSET-SUM instance says 'Yes' *iff* the corresponding PARTITION instance says 'Yes'

Satisfiability reduces to Independent Set

SAT $\leq_p$ INDEPENDENT-SET

Given an instance I of SAT, we construct an instance (G, k) of INDEPENDENT-SET that has an independent set of size k iff I is satisfiable.

Example: $(\overline{x} \vee y \vee \overline{z}) \wedge (x \vee \overline{y} \vee z) \wedge (y \vee \overline{x})$.

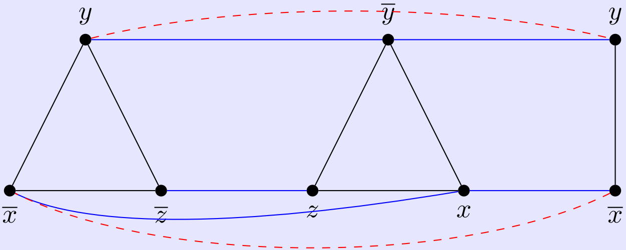


Satisfiability reduces to Independent Set

SAT $\leq_p$ INDEPENDENT-SET

Given an instance I of SAT, we construct an instance (G, k) of INDEPENDENT-SET that has an independent set of size k iff I is satisfiable.

Example: $(\overline{x} \vee y \vee \overline{z}) \wedge (x \vee \overline{y} \vee z) \wedge (y \vee \overline{x})$.



Satisfiability reduces to 3-Satisfiability

SAT

$C_1 = \overline{x_1}$
$C_2 = x_1 \vee x_2$
$C_3 = x_1 \vee \overline{x_2} \vee x_3$
$C_4 = \overline{x_4} \vee x_5 \vee x_6 \vee x_7$

$C_5 = \overline{x_3} \vee \overline{x_5} \vee x_1 \vee x_7 \vee \overline{x_2}$

Satisfiability reduces to 3-Satisfiability

SAT

$C_1 = \overline{x_1}$
$C_2 = x_1 \vee x_2$
$C_3 = x_1 \vee \overline{x_2} \vee x_3$
$C_4 = \overline{x_4} \vee x_5 \vee x_6 \vee x_7$

$C_5 = \overline{x_3} \vee \overline{x_5} \vee x_1 \vee x_7 \vee \overline{x_2}$

3-SAT

$C_{10} = \overline{x_1} \vee y_1 \vee y_1$

Satisfiability reduces to 3-Satisfiability

SAT

$C_1 = \overline{x_1}$
$C_2 = x_1 \vee x_2$
$C_3 = x_1 \vee \overline{x_2} \vee x_3$
$C_4 = \overline{x_4} \vee x_5 \vee x_6 \vee x_7$

$C_5 = \overline{x_3} \vee \overline{x_5} \vee x_1 \vee x_7 \vee \overline{x_2}$

3-SAT

$C_{10} = \overline{x_1} \vee y_1 \vee y_1$
$C_{20} = x_1 \vee x_2 \vee y_2$

Satisfiability reduces to 3-Satisfiability

SAT

$C_1 = \overline{x_1}$
$C_2 = x_1 \vee x_2$
$C_3 = x_1 \vee \overline{x_2} \vee x_3$
$C_4 = \overline{x_4} \vee x_5 \vee x_6 \vee x_7$

$C_5 = \overline{x_3} \vee \overline{x_5} \vee x_1 \vee x_7 \vee \overline{x_2}$

3-SAT

$C_{10} = \overline{x_1} \vee y_1 \vee y_1$
$C_{20} = x_1 \vee x_2 \vee y_2$
$C_{30} = x_1 \vee \overline{x_2} \vee x_3$

Satisfiability reduces to 3-Satisfiability

SAT

$C_1 = \overline{x_1}$
$C_2 = x_1 \vee x_2$
$C_3 = x_1 \vee \overline{x_2} \vee x_3$
$C_4 = \overline{x_4} \vee x_5 \vee x_6 \vee x_7$

$C_5 = \overline{x_3} \vee \overline{x_5} \vee x_1 \vee x_7 \vee \overline{x_2}$

3-SAT

$C_{10} = \overline{x_1} \vee y_1 \vee y_1$
$C_{20} = x_1 \vee x_2 \vee y_2$
$C_{30} = x_1 \vee \overline{x_2} \vee x_3$
$C_{40} = \overline{x_4} \vee x_5 \vee y_3$
$C_{41} = \overline{y_3} \vee x_6 \vee x_7$

Satisfiability reduces to 3-Satisfiability

SAT

$C_1 = \overline{x_1}$
$C_2 = x_1 \vee x_2$
$C_3 = x_1 \vee \overline{x_2} \vee x_3$
$C_4 = \overline{x_4} \vee x_5 \vee x_6 \vee x_7$

$C_5 = \overline{x_3} \vee \overline{x_5} \vee x_1 \vee x_7 \vee \overline{x_2}$

3-SAT

$C_{10} = \overline{x_1} \vee y_1 \vee y_1$
$C_{20} = x_1 \vee x_2 \vee y_2$
$C_{30} = x_1 \vee \overline{x_2} \vee x_3$
$C_{40} = \overline{x_4} \vee x_5 \vee y_3$
$C_{41} = \overline{y_3} \vee x_6 \vee x_7$
$C_{50} = \overline{x_3} \vee \overline{x_5} \vee y_4$
$C_{51} = \overline{y_4} \vee x_1 \vee y_5$
$C_{52} = \overline{y_5} \vee x_7 \vee y_6$
$C_{53} = \overline{y_6} \vee \overline{x_2} \vee y_7$

Exercise

1. 3-Satisfiability reduces to Directed Hamiltonian Cycle
2. 3-Satisfiability reduces to Longest Path Problem
3. Hamiltonian Cycle reduces to Traveling Saleman Problem
4. 3-Satisfiability reduces to Subset-Sum