Advanced Core in Algorithm Design #11
算法設計要論 第 11 回

Yasushi Kawase
河瀬 康志

Dec. 21th, 2021

last update: 10:09am, December 21, 2021

Schedule

Lec. #	Date	Topics
1	10/5	Introduction, Stable matching
2	10/12	Basics of Algorithm Analysis, Graphs
3	10/19	Greedy Algorithms (1/2)
4	10/26	Greedy Algorithms (2/2)
5	11/2	Divide and Conquer (1/2)
6	11/9	Divide and Conquer (2/2)
7	11/16	Dynamic Programming (1/2)
8	11/30	Dynamic Programming (2/2)
9	12/7	Network Flow (1/2)
10	12/14	Network Flow (2/2)
11	12/21	NP and Computational Intractability
12	1/4	Approximation Algorithms (1/2)
13	1/11	Approximation Algorithms (2/2)
14	1/18	Final Examination

Outline

1. Polynomial-Time Reductions
2. P vs NP
3. NP-completeness

Problem, Algorithm, Running time

- A computational problem can be viewed as
  a map $f\colon I \to S$ from the set of instances (I) to the set of solutions (S)

Primality testing $I = \mathbb{N}$, $S = \{\texttt{yes}, \texttt{no}\}$, $f(1) = \texttt{no}$, $f(2) = \texttt{yes}$, $f(3) = \texttt{yes}$, $f(4) = \texttt{no}, \ldots$

- An algorithm for computing f is a set of rules such that
  by following them we can compute $f(x)$ given any input $x \in I$

- An algorithm for computing f is said to be $T(n)$-time if
  it outputs $f(x)$ in at most $T(|x|)$ steps for any $x \in I$

  ($|x|$: length of x)

Polynomial-time algorithms

Definition

$p(n) = \mathrm{O}(n^c)$ for some $c > 0$

Polynomial-time algorithm is $p(n)$-time algorithm for some polynomial p

"Efficient" algorithm $\iff$ polynomial-time algorithm

Example: max-flow $(G = (V, E), s, t, c\colon E \to \mathbb{Z}_{++})$

- size of an instance is $\mathrm{O}(|V| + |E| + \sum_{e \in E} \log c(e))$
- Ford–Fulkerson: $\mathrm{O}(|E| \sum_{e \in E} c(e))$ time → not polynomial-time
- Capacity scaling: $\mathrm{O}(|E|^2 \log \max_{e \in E} c(e))$ time → (weakly) polynomial-time
- Edmonds–Karp: $\mathrm{O}(|E|^2 |V|)$ time → (strongly) polynomial-time

Classify problems

we want to classify tractable problems

Definitoin

A problem is polynomial-time solvable if ∃polynomial-time alg. for it

Poynlmial-time solvable

- shortest path
- min cut
- bipartite matching
- linear programming
- primality testing

Probably not

- longest path
- max cut
- 3-dimensional matching
- integer linear programming
- factoring

Polynomial-time reductions

Definition

Problem X is polynomial-time reducible to problem Y ($X \leq_{\mathrm{P}} Y$) if arbitrary instances of problem X can be solved using:

- polynomial number of standard computational steps
- polynomial number of calls to oracle that solves problem Y

Example Bipartite matching $\leq_{\mathrm{P}}$ Max-flow

Observations

- $X \leq_{\mathrm{P}} Y$ and Y is solvable in poly-time $\Longrightarrow$ X is solvable in poly-time
- $X \leq_{\mathrm{P}} Y$ and X is not solvable in poly-time $\Longrightarrow$ Y is not solvable in poly-time
- $X \leq_{\mathrm{P}} Y$ and $Y \leq_{\mathrm{P}} Z \Longrightarrow X \leq_{\mathrm{P}} Z$ (transitivity)

Outline

1. Polynomial-Time Reductions
2. P vs NP
3. NP-completeness

Decision problem

Definition: Decision problem

- a problem where the answer for every instance is either yes or no
- can be represented as a map from $\{0,1\}^*$ to $\{0,1\}$

$\bigcup_{n\geq 0}\{0,1\}^n$ no yes

- simple encodings can be used to represent general objects

integers, pairs of integers, graphs, vectors, matrices,...

- $L_f = \{x \mid f(x) = 1\} \subseteq \{0,1\}^*$ is called language

Example: primality testing (determining whether an input number p is prime)

- $f(x) = 1$ iff x is a representation of a prime
- $f(1) = 0$, $f(11) = 1$, $f(101) = 1$

Uncomputable decision problem

Theorem

$\exists$ decision problem that is not computable by any algorithm

- The number of decision problems is uncountable
- The number of algorithm is countable

P and NP

Definition: class P

The set of decision problems for which ∃poly-time algorithm

Definition: class NP

The set of decision problems f for which $\exists g$ such that

- g is computable by a polynomial-time algorithm
- $f(x) = 1 \iff \exists w, \exists polynomial \mathsf{p},\ |w| \leq p(|x|) and g(x, w) = 1$

witness

- **P** stands for Polynomial-time
- **NP** stands for Non-deterministic Polynomial-time
- Observation: $\mathbf{P} \subseteq \mathbf{NP}$

$\mathbf{P}$ vs $\mathbf{NP}$

Conjecture

$\mathbf{P} \neq \mathbf{NP}$

- Most computer scientists believe that $\mathbf{P} \neq \mathbf{NP}$
- \$1,000,000 for resolution of $\mathbf{P}$ vs $\mathbf{NP}$ problem (millennium prize)

  `https://www.claymath.org/millennium-problems/p-vs-np-problem`

  1. Yang–Mills and Mass Gap
  2. Riemann Hypothesis
  3. P vs NP Problem
  4. Navier–Stokes Equation
  5. Hodge Conjecture
  6. Poincaré Conjecture → solved by Grigori Perelman
  7. Birch and Swinnerton-Dyer Conjecture

Problems in **NP** (1/4)

Satisfiability problem (SAT)

Given a CNF formula Φ, does it have a satisfying truth assignment?

3-SAT

SAT where each clause contains exactly 3 literals

- boolean variables: $x_1, \ldots, x_n$
- literal: $x_1, \ldots, x_n, \overline{x_1}, \ldots, \overline{x_n}$
- clause: a disjunction of literals, e.g., $C_j = x_1 \vee \overline{x_2} \vee x_3$
- conjunctive normal form : conjunction of clauses, e.g., $\Phi = C_1 \wedge C_2 \wedge C_5$

  CNF

Examples

- $\Phi = (\overline{x_1} \vee x_2 \vee x_3) \wedge (x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee x_2 \vee x_4)$ ⟶ Yes
- $\Phi = (x_1 \vee x_2) \wedge (x_1 \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2})$ ⟶ No

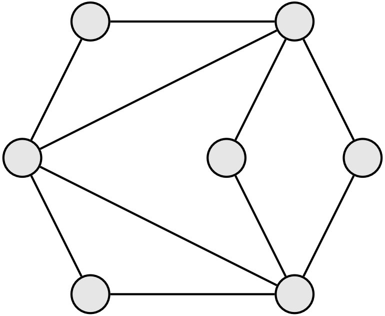
Problems in $\mathbf{NP}$ (2/4)

Independent set problem (IS)

Given a graph $G = (V, E)$ and an integer k, is there $S \subseteq V$ such that $|S| \geq k$ and no two vertices in S are adjacent?

Example

$k = 4$

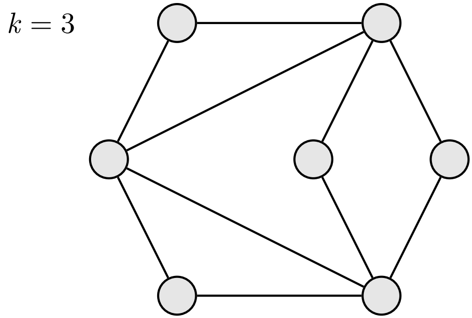


Problems in $\mathbf{NP}$ (3/4)

Vertex cover (VC)

Given a graph $G = (V, E)$ and an integer k, is there $C \subseteq V$ such that $|C| \leq k$ and each edge is incident to at least one vertex in C?

Example



Problems in $\mathbf{NP}$ (4/4)

Set cover problem (Set-Cover)

Given a set U of elements, $S_1, S_2, \ldots, S_m \subseteq U$, an integer k, is there $J \subseteq \{1, 2, \ldots, m\}$ such that $|I| \leq k$ and $\bigcup_{j \in J} S_j = U$?

Example

- $U = \{1, 2, 3, 4\}$
- $S_1 = \{1, 3\}$
- $S_2 = \{1, 2\}$
- $S_3 = \{2, 3, 4\}$
- $k = 2$

Outline

1. Polynomial-Time Reductions
2. P vs NP
3. NP-completeness

$\mathbf{NP}$-complete

Definition

- A problem X is $\mathbf{NP}$-hard if $Y \leq_{\mathrm{P}} X$ for every $Y \in \mathbf{NP}$
- A problem X is $\mathbf{NP}$-complete if it is $\mathbf{NP}$-hard and in $\mathbf{NP}$

Proposition

- If X is $\mathbf{NP}$-hard and $X \leq_{\mathrm{P}} Y$, then Y is also $\mathbf{NP}$-hard
- If X is $\mathbf{NP}$-complete and $X \leq_{\mathrm{P}} Y \in \mathbf{NP}$, then Y is also $\mathbf{NP}$-complete
- If X is $\mathbf{NP}$-complete, then $X \in \mathbf{P}$ iff $\mathbf{P} = \mathbf{NP}$

Q: are there any "natural" $\mathbf{NP}$-complete problems?

The first $\mathbf{NP}$-complete problem

Cook–Levin Theorem

SAT is $\mathbf{NP}$-complete

Proof sketch formal proof requires nondeterministic Turing machine

- We show $X \leq_{\mathrm{P}}$ SAT for any $X \in \mathbf{NP}$
- Let g be a certificate of X
  - g is computable by a polynomial-time algorithm
  - $f(x) = 1 \iff \exists w,\ |w| \leq p(|x|),\ g(x, w) = 1$
- We construct a CNF that "simulates" the algorithm
  - the algorithm for g runs in poly-space and poly-step
  - make a boolean variable for every pair of place and step

SAT reduces to 3-SAT

Theorem

SAT $\leq_{\mathrm{P}}$ 3-SAT, and hence 3-SAT is $\mathbf{NP}$-complete

Proof

- Transform each clause individually
  - $C = \ell \longrightarrow (\ell \vee z_1 \vee z_2) \wedge (\ell \vee z_1 \vee \overline{z_2}) \wedge (\ell \vee \overline{z_1} \vee z_2) \wedge (\ell \vee \overline{z_1} \vee \overline{z_2})$
  - $C = \ell_1 \vee \ell_2 \longrightarrow (\ell_1 \vee \ell_2 \vee z) \wedge (\ell_1 \vee \ell_2 \vee \overline{z})$
  - $C = \ell_1 \vee \ell_2 \vee \ell_3 \longrightarrow \ell_1 \vee \ell_2 \vee \ell_3$
  - $C = \ell_1 \vee \ell_2 \vee \cdots \vee \ell_k \ (k > 3)$
    $\longrightarrow (\ell_1 \vee \ell_2 \vee z_1) \wedge (\ell_3 \vee \overline{z_1} \vee z_2) \wedge (\ell_4 \vee \overline{z_2} \vee z_3) \wedge \cdots \wedge (\ell_{k-2} \vee \overline{z_{k-4}} \vee z_{k-3}) \wedge (\ell_{k-1} \vee \ell_k \vee \overline{z_{k-3}})$
- The reduction preserves satisfiability

3-SAT reduces to Independent set problem

Theorem

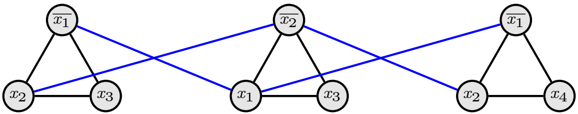
3-SAT $\leq_{\mathrm{P}}$ IS, and hence IS is $\mathbf{NP}$-complete

Proof

- Given a 3-SAT instance Φ, we construct an IS instance (G, k) as follows
  - Each clause ⟶ triangle (3 vertices and 3 edges)
  - Connect literal to each of its negations
  - $k = |\Phi|$
- Φ is satisfiable $\iff$ G has an independent set of size k

Example $\Phi = (\overline{x_1} \vee x_2 \vee x_3) \wedge (x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee x_2 \vee x_4)$

$k = 3$



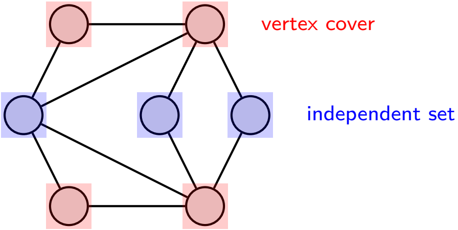
Vertext cover problem

Theorem

IS $\leq_{\mathrm{P}}$ VC, and hence VC is $\mathbf{NP}$-complete

Proof

- Observation: S is an independent set $\iff$ $V \setminus S$ is a vertex cover
- (G, k) is a yes-instance of IS $\iff$ $(G, |V| - k)$ is a yes-instance of VC



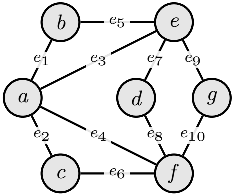
Set cover problem

Theorem

VC $\leq_{\mathrm{P}}$ Set-Cover, and hence Set-Cover is $\mathbf{NP}$-complete

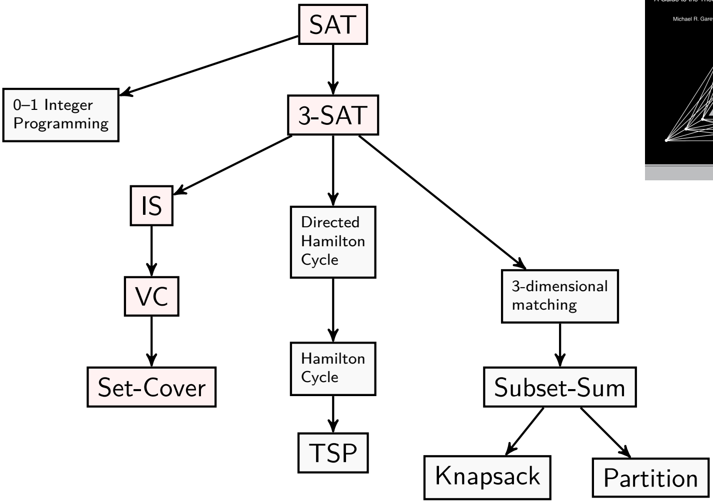
Proof

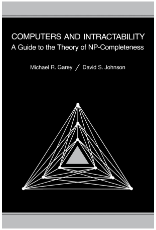
- Given a VC instance (G, k), we construct (U, S, k') as follows
  - $U = E$, $k' = k$
  - For each $v \in V$, $S_v = \{e \in E \mid e \text{ incident to } v\}$
- G has a vertex cover of size $k \iff (U, S)$ has a set cover of size k



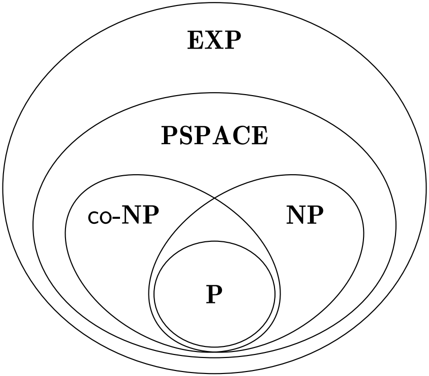
- $U = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$
- $S_a = \{1, 2, 3, 4\}$, $S_b = \{1, 5\}$, $S_c = \{2, 6\}$, $S_d = \{7, 8\}$, $S_e = \{3, 5, 7, 9\}$, $S_f = \{4, 6, 8, 10\}$, $S_g = \{9, 10\}$

Basic $\mathbf{NP}$-complete problems





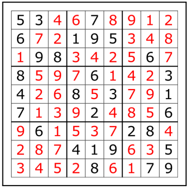
Other Basic Complexity Classes



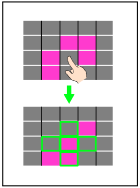
- $\mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{PSPACE} \subseteq \mathbf{EXP}$
- $\mathbf{P} \neq \mathbf{EXP}$
- cf. `https://complexityzoo.net/Complexity_Zoo` (545 classes)

Quiz

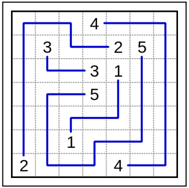
Which puzzles are known to be NP-hard?



$n \times n$ sudoku



$n \times n$ lights out



numberlink

$n \times n \times n$ Rubik's cube