# NP-Completeness Theory

Previously:

- Positive results: design efficient algorithm to solve problems

This part:

- Negative results: show some problems cannot be solved efficiently.

▷ Why study negative results?

- Practical: avoid wasting time on designing algs
- Theoretical:
  - THE first problem of computer science [Turing '1936]
  - Leads to better algorithms
  - fun!

## ▷ Preliminary

- Computation model
  - Finite automata. Turing Machine. Quantum Computer.
  - RAM model

- Decision problem vs. Optimization problem.
  - Decision problem: output is 0/1. (single bit)
    - Exmp: "Is there a $(s,t)$-path in the graph $G$ of length $\le L$?"
  - Optimization problem: output is arbitrary
    - Exmp: "What is the shortest $(s,t)$-path length in graph $G$?"

- Optimization $\Longrightarrow$ Decision
  - Trivial: E.g. if you know the shortest (s,t)-path length, then surely you can answer the question "Is it $\leq L$?"

- Decision $\Longrightarrow$ Optimization
  - Binary Search: Guess a value $L$, then solve the decision prob "Is it $\leq L$?"
    - If "Yes", then try a smaller $L$
    - Otherwise, try a larger $L$.
  - Incurs an extra $O(\log n)$ factor on running time.

- "Efficient" $\equiv$ polynomial time, i.e. $n^{O(1)}$

- But why? a $O(n^{1000})$-time alg doesn't seem to be useful.

- Ans:

  - For most realworld problems, if $\exists$ poly-time alg. then often the running time is $O(n^k)$ for some $k \leq 4$

  - A "Dichotomy phenomenon": often for natural problemes
    - either we have a $O(n^k)$-time alg with some small $k$
    - or the best alg we have runs in $2^{O(n)}$ time

## ▷ Some hard problems

- Hamiltonian Path:

> Input: graph $G=(V,E)$, and $s,t \in V$
>
> Output: find a $(s,t)$-path that visits every vertex in $V$ exactly once.

- Max Indep Set:

> Input: graph $G=(V,E)$
>
> Output: indep set $S \subseteq V$ with maximum size.

▷ P : polynomial-time solvable problems.

Def: The complexity class P is the set of (decision) problems that can be solved in polyn time.

- Exmp : MST, shortest-path, interval scheduling

- Knapsack is not known to be in P

▷ NP: Polynomial-time certifiable problems.

- Exmp: Hamiltonian Path prob. "Is there a $(s,t)$-HP in $G$?"
  - Certification:
    - Given: Input graph $G$, and a $(s,t)$-path $P$
    - Goal: verify if $P$ is a HP
    - Easy to solve.
  - $P$ is a certificate for the answer "Yes"
  - Alg to certify $P$ runs in poly(n) time.

  ↳ certifier.

Def: The complexity class NP is the set of (decision) problems that can be certified in polyn time.

Exmp: Hamiltonian Path, Max Indep Set, Vertex Cover. Knapsack.

- Max Indep Set: "Is the Max Indep Set has size $\geq L$?"
  - Certificate: an indep set of size $\geq L$
  - Certifier: Count the size of the set & check if $\forall u, v \in$ the given set. $(u,v) \notin E$

- Observation: $P \subseteq NP$.

## ▷ The "P vs NP" problem

$\boxed{\text{Question: Is } P = NP\text{ ?}}$

- Common belief: $P \neq NP$
  - "Assume $P \neq NP$, then HP doesn't have poly-time alg"
- Consequence:
  - If $P \neq NP$ : not much will change.
  - If $P = NP$ : If one can check a sol efficiently, then one can find a sol efficiently.

▷ Reduction:

- How do we compare the difficulty of two problems when we don't know the answer of "$P \stackrel{?}{=} NP$".

E.g. How do we know MST is easier than HP, given that we cannot prove there's no poly-time alg for HP?

Reduction: If an alg A for prob X can be used to solve prob Y, then X is at least as hard as Y.

Consider two computation problems X, Y; if for any instance of problem Y, we can convert it to an instance of X in poly-time, then we say Y is poly-time reducible to X, denoted as $Y \le_p X$.

Corollary (positive):

Given a poly-time alg A that solves X. if $Y \le_p X$, then Y can also be solved in poly-time.

**Corollary (negative)**

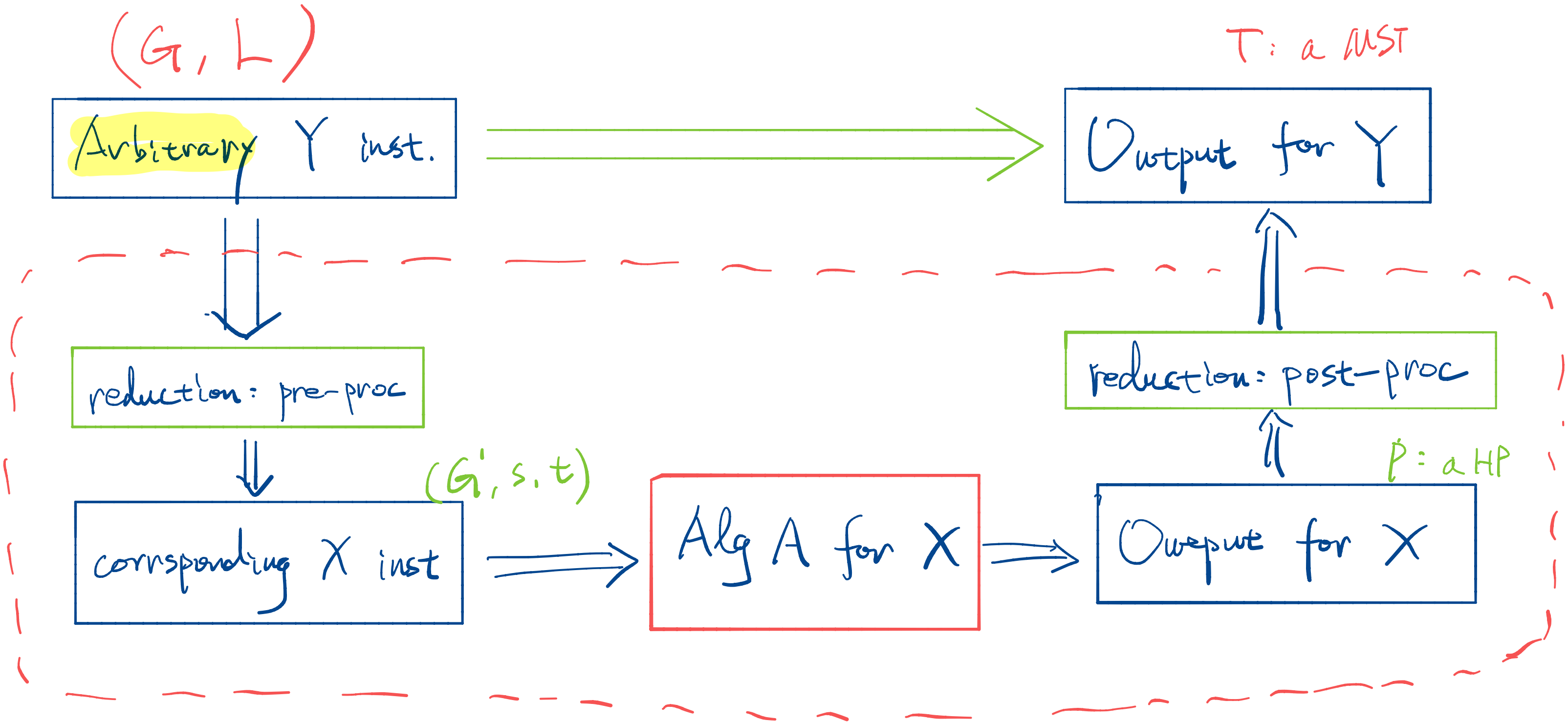
If $Y \leq_p X$ and $Y$ cannot be solved in poly-time, then $X$ cannot be solved in poly time either.

i.e. $Y \leq_p X$ implies $X$ is at least as hard as $Y$.

Exmp:

Suppose we can show $MST \leq_p HP$, then at least we know $HP$ is no easier than $MST$.

▷ Reduction : Converting Y to X



To show $Y \leq_p X$:

- Focusing on decision problem.
- Design a reduction alg that:

① show an "Yes" inst of Y implies a "Yes" inst of X

② show an "No" inst of Y implies a "No" inst of X

– Or equivalently: show a "Yes" inst of X implies a "Yes" inst of Y

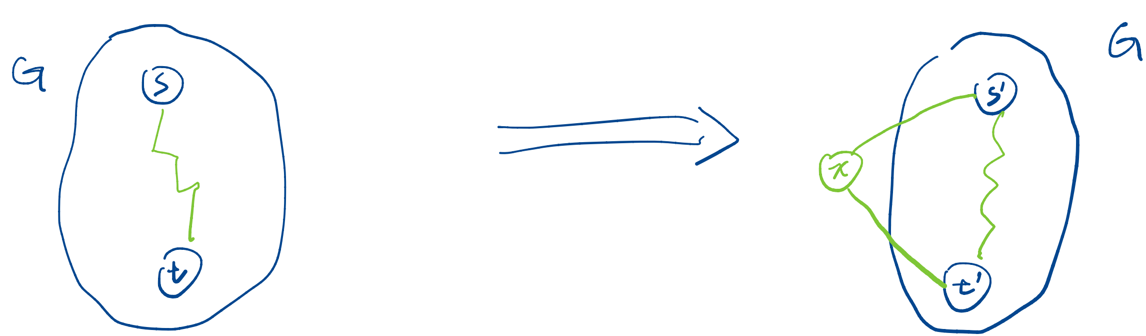
Exmp: HP $\le_p$ HC

**Hamiltonian Cycle (HC)**

Input: $G = (V, E)$, $s \in V$

Output: whether there is a cycle that starts from $s$, visits every other vertex exactly once, and ends at $s$.

HP inst:                                        HC inst



## ▷ NP-Complete problem.

A problem $X$ is NP-Complete if

① $X \in NP$

② $\forall Y \in NP, \ Y \leq_p X$

- $X$ is "the hardest problem" in NP
- Positive: A poly-time alg for $X$ will imply $P = NP$
- Negative: If you believe $P \neq NP$, and proved a problem to be NPC, then you can give up on designing poly-time Algs for it.

Exmp: HP, HC, Max Indep Set, Vertex Cover.

Knapsack, Subset-Sum.

"Dichotomy":

Most natural NP problems are either NPC or P

(There do exist some "intermediate" problems, e.g.

Factoring; Graph Isomorphism)

▷ The first NPC problem: 3SAT

**Def (3-CNF):**

- Boolean vbl $x_1, \cdots, x_n \in \{\text{True}, \text{False}\}$
- Literals: $x_i$ or $\neg x_i$
- Clause: disjunction ("or") or $\leq 3$ literals

  e.g. $x_3 \vee \neg x_4$, $x_1 \vee x_8 \vee x_9$. $\neg x_2 \vee \neg x_5$
- 3-CNF formula: conjunction ("and") of clauses:

  $$(x_1 \vee x_2 \vee \neg x_3) \wedge (x_2 \vee x_4 \vee x_5) \wedge (\neg x_1 \vee \neg x_3 \vee x_4)$$

Prob (3-SAT)

Input : A 3-CNF formula involve vbl $x_1, \cdots, x_n$

Output : Whether the 3-CNF is satisfiable, i.e

$\exists$ an assignment from $\{x_1, \cdots, x_n\}$ to $\{\text{True}, \text{False}\}$

s.t. the 3-CNF evaluates to True.

- "Satisfied" : ① every clause evals to True

  ② In each clause, at least one literal evals to true.

- $(x_1 \vee x_2 \vee \neg x_3) \wedge (x_2 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee \neg x_3 \vee x_4)$

  T / / / / T / / T

- Thm [Cook' 1970s] : 3-SAT is NPC

▷ 3SAT $\le_p$ Indep Set. (i.e. Indep Set is NPC)

Indep Set inst:

$(G, k)$

"Is there an indep set in G of size $\geq k$?"

Exmp: 3SAT inst

$$\phi = (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_2 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee \neg x_3 \vee x_4)$$

Goal: construct an IndepSet inst $(G, k)$ based on $\phi$ s.t.

① $\phi$ is satisfiable $\Longrightarrow$ $\exists$ indep set of size $\geq k$ in G

② $\phi$ is unsatisfiable $\Longrightarrow$ $\forall$ indep set in G is of size $< k$

or equivalently:

$\exists$ indep set of size $\geq k$ in G $\Longrightarrow$ $\phi$ is satisfiable

3-SAT

$x_1 \leftarrow T$ , $x_2 \leftarrow T$

$x_4 \leftarrow T$ , $x_3 \leftarrow F$

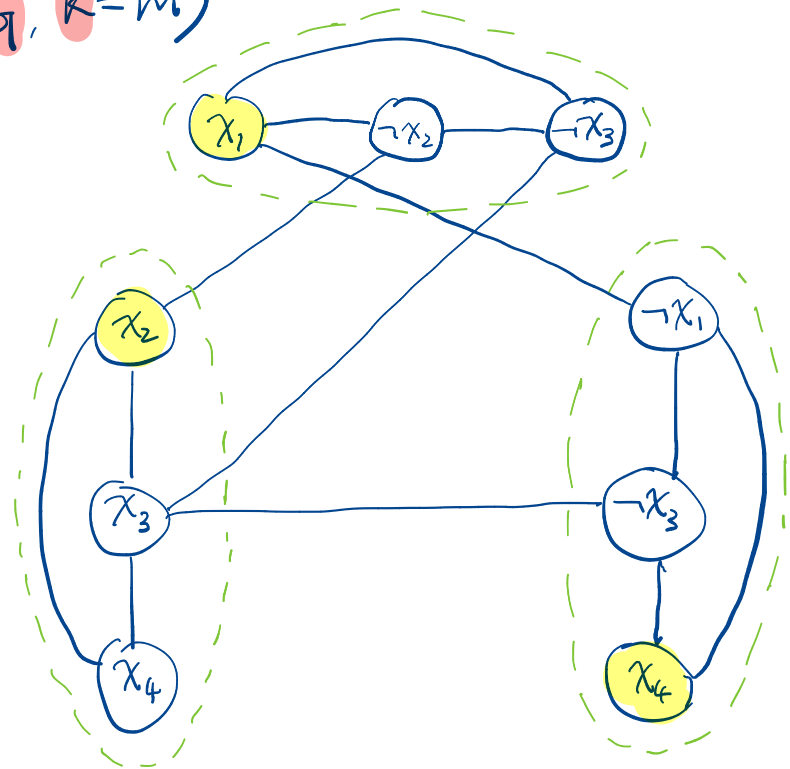
$$\phi = (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_2 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee \neg x_3 \vee x_4)$$

Let $n :=$ #vbls in $\phi$

$m :=$ #clauses

Indep Set

$(G, k=m)$




clauses $\iff$ group.

① $\phi$ is satisfiable $\Longrightarrow$ $\exists$ indep set $S$ of size $\geq k = m$ in $G$

- Every clause has $\geq 1$ literal is true
- Pick exactly one literal that's true, add the corresponding vertex in $G$ to a set $S$

$\boxed{\text{Claim: } S \text{ is a size-}m \text{ indep set in } G.}$

② $\exists$ indep set $S$ of size $\geq k = m$ in $G$ $\Longrightarrow$ $\phi$ is satisfiable

$\boxed{\text{Obs: } S \text{ contain exactly one vertex of each group}}$

- For each $v \in S$, let the corresponding literal be true,

$\boxed{\text{Claim: above assignment satisfies } \phi.}$

▷ Indep Set $\leq_p$ Vertex Cover (VC)

**Prob** Vertex Cover (VC):

Input: graph $G = (V, E)$, integer $k$

Output: whether there's a VC of $G$ of size $\leq k$

( Recall: a VC of $G$ is some $S \subseteq V$ s.t. $\forall (u,v) \in E$, at least one of $u, v$ are in $S$. )

**Obs**: If $S \subseteq V$ is a VC in $G$, then $V \setminus S$ is an IS in $G$

Pf: Suppose $V \setminus S$ is not an IS. Then, $\exists u, v \in V \setminus S$, s.t. $(u,v) \in E$. However, since $S$ is a VC, at least one of $u, v$ are in $S$, which contradicts with $u, v \in V \setminus S$. $\square$

▷ Pf of IndepSet $\leq_p$ Vertex Cover

- $\forall$ IS inst. $(G, k)$ with $|V| = n$

$\xrightarrow{\text{reduction}}$ VC inst $(G, n-k)$, then

① $\exists$ IS of size $\geq k$ $\Longrightarrow$ $\exists$ VC of size $\leq n-k$

- Trivial by the previous Obs: take the complement of the IS

② $\exists$ VC of size $\leq n-k$ $\Longrightarrow$ $\exists$ IS of size $\geq k$

- Same as above: take the complement of the VC.

## ▷ Dealing with NP-hard problems

**Def:** Problem $X$ is NP-hard if $\forall Y \in NP$, $Y \leq_p X$

Note: $X$ itself is not required to be in NP.

- All NPC problems are NP-hard, but not vice versa.
- No hope for poly-time alg (assuming $P \neq NP$)? :
  - Faster exp-time alg
  - Approx alg
  - Solving for special cases

▷ Faster exp-time alg

3-SAT:

- Brute-force: $O(2^n \cdot poly(n))$
- $2^n \longrightarrow 1.844^n \longrightarrow 1.34^n$
- In practice: up to 10 000 vbls can be solved in short time

Traveling Salesman Problem (i.e. the shortest Hamiltonian Cycle)

- Brute-force: $O(n! \cdot poly(n)) \longrightarrow O(2^n \cdot poly(n))$
- In practice: Euclidean TSP with ~ 100 000 vertices.

▷ Approx Alg.

Idea: ask for sub-optimal sols that are guaranteed not too far away from opt.

Exmp:

| |
|---|
| Prob (Min Vertex Cover): given graph G. find the Vertex Cov in G of minimum size. |

- Can we find an VC of size $\leq \alpha \cdot OPT$ for some small $\alpha \geq 1$?

  Ans: there exists an alg outputs $VC \leq 2 \cdot OPT$ (2-approx)

- Can we find an IS of size $\geq \alpha \cdot OPT$ for some big $\alpha < 1$?

  Ans: unfortunately, no. No polytime alg can achieve $\alpha > \Omega(\frac{1}{n})$

▷ Solving for special cases

Exmp: Max-Indep Set is polytime-solvable on

- Trees
- Bounded-treewidth graph
- Interval graphs

...