CMSC 451 Nov. 9

NP-Completeness

Lectures before: How to solve problems use efficient algorithms.

Efficient: polynomial time e.g. $O(n)$, $O(n^3)$ $\cdot \Theta(n \log n)$

Hard : exponential time e.g. 2^n, ...

Easy problems vs. Hard problem.

Minimum Spanning Tree (MST):

Find smallest weighted tree to connect all vertices

- has efficient algorithms.

Traveling Salesman Problem (TSP):

Find smallest weighted path to connect all vertices

3-coloring, independent set, hamiltonian path.

Q: How do we characterize hardness of a problem?

"easy" means "has polynomial-time algorithm"

"hard" --- ?

has no polynomial time algo.

↓

NP-Completeness Theory in early 1970s.

Decision Problems: Yes / No. ← circulation

<=> Optimization: Find the best ← max flow

For any optimization prob., there is a decision ver. of it.

Decision ver. of MST:

Is the graph G has a spanning tree w/ weight $\leq s$?

Decision ver. is easier than original problem

Language Recognition: → notations from 1970s.

Language is a set of string → aba ←

eg. set of palindromes, ...

Language Recognition: given s & L, tell whether $s \in L$.

s: string; L: language

Every decision problem (can be encoded) is a Language Recognition problem.

$MST = \{ \text{serialize}(G, s) \mid G \text{ has a spanning tree of weight} \leq s \}$

(G, s) is yes in MST, $\Leftrightarrow (G, s) \in MST$

P & NP.

P: languages that can be recognized in polynomial time

$MST \in P \Rightarrow$ MST has a poly. time algo.

NP: languages that can be verified in polynomial time.

Example: Hamiltonian cycle (HC):

Given a graph G, whether or not $\exists$, a cycle s.t. every vertex is visited exactly once.

Solve the problem: hard (up to now)

Verify the problem: easy, just show the cycle.

Formal Definition of NP:

Language $L \in NP$, if,

exist verification algo (poly-time) V,

$s \in L \Leftrightarrow \exists\, c$, s.t. $(s, c) \in V$.

HC: $G \in HC \Leftrightarrow \exists\, c$, $(G, c) \in V_{HC}$ ← check whether c visits all vertices exactly once.

Unique HC: Whether a graph G has a unique Hamiltonian cycle.

⇒ hard to verify!

not proved to be in NP!

Reduction:

How to define hard?

Intuition: Define "harder" ⇒ hard = "hardest".

B is easier than A

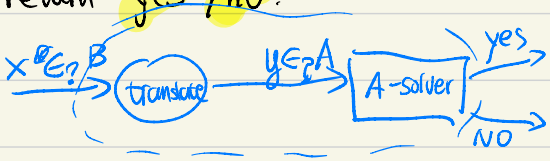
A is harder than B ⇒

if A is solvable, B is also solvable.

⇒ ∃ Algorithm to translate B instances into A instances.

solve B:

1. translate into an A instance
2. call A's algorithm
3. return yes / no.



Example:

Last Lecture: circulation → max flow

circulation w/ lower bound → circulation

want to prove prob A is hard, reduce another known hard prob. B into it.

Proof contains:

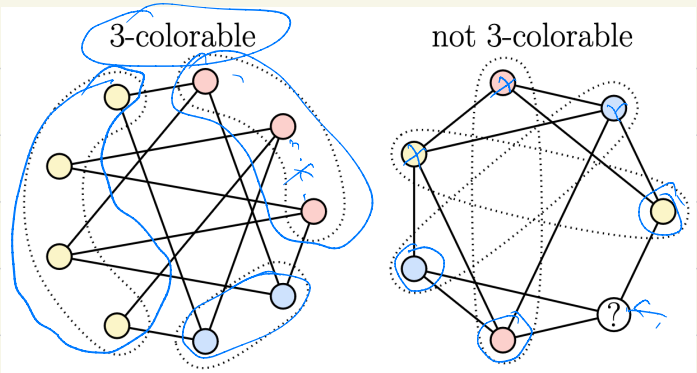
1. a translation algorithm. (in poly-time)
2. $x \in A \iff f(x) \in B$

$\Rightarrow$ A is hard $\Rightarrow$ B is hard.

3-coloring (3Col):

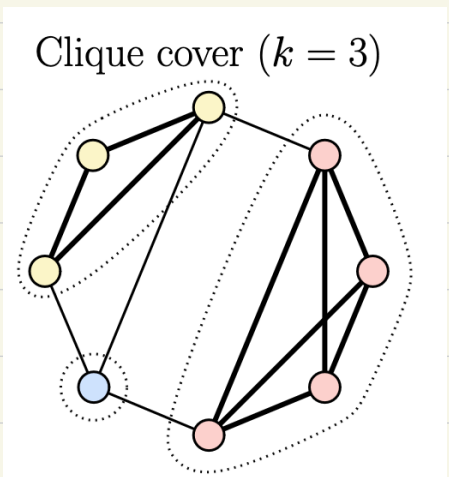
Color (all vertices of) a graph G into 3 colors s.t. no edges between vertices w/ same color

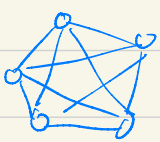
- Hard problem !



Clique Cover (CCov).

Given graph G & number k, partition vertices into k sets, s.t. each set is a clique ← all vertices are adjacent





Want to show CCov is hard. How?

- Reduce a known-hard problem to it.
- e.g. $3Col \leq_p CCov$.

Proof: Translation algorithm:

complement graph

Consider the graph $\bar{G}$

$(u,v) \in G \Rightarrow (u,v) \notin \bar{G}$

$(u,v) \notin G \Rightarrow (u,v) \in \bar{G}$.

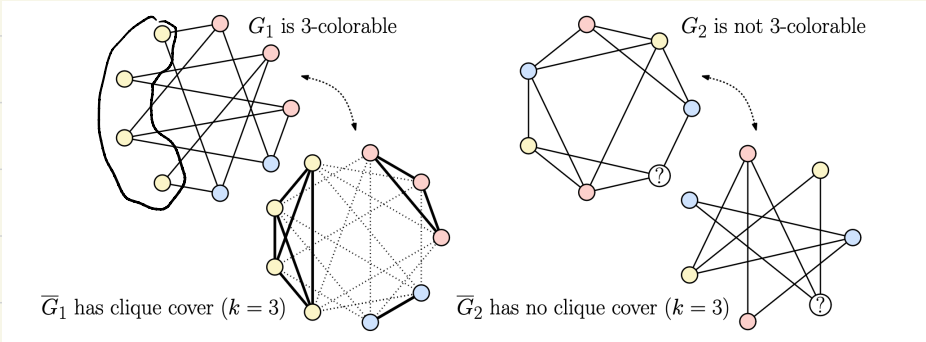
and set $k = 3$.

- also need to show the translation is poly-time.

Equivalence Proof:

① $G \in 3Col \Rightarrow (\bar{G}, 3) \in CCov$.

② $(\bar{G}, 3) \in CCov \Rightarrow G \in 3Col$.



Need to have a written proof!

Definition: L_1 is polynomial-time reducible to L_2, ($L_1 \leq_p L_2$)
if there is a polynomial-time computable function f,
s.t. $\forall x$, $x \in L_1 \iff f(x) \in L_2$.

Language L is NP-hard if

$\forall L' \in NP$, $L' \leq_p L$.

L is NP-complete if.

$L \in NP$ and L is NP-hard.

How is it possible? (Next Lecture)