CMPSC 465
Data Structures and Algorithms
Spring 2022

Instructor: Chunhao Wang

NP and Computational Hardness

NP and Computational Hardness

Polynomial-time reduction
(Kleinberg-Tardos, Section 8.1, 8.2)

Motivation

Which problem is harder?

- Problem A: it takes me a week to come up with an $O(n^2)$ algorithm
- Problem B: It's straightforward to design a brute-force algorithm with running time $O(2^n)$, but it's the best-known algorithm

Is Problem B really hard? How do we prove hardness?

Proving hardness

We can prove **lower bound** for some problems. For example: for sorting $\Omega(n \log n)$

For hard problems, can we get something like $\Omega(2^n)$?
Unfortunately, for most hard problems, we can't either find an $O(\mathrm{poly}(n))$ time algorithm or prove a lower bound like $\Omega(\exp(n))$

Instead, we classify hard computational problems. We prove they are "equivalent" in the sense that

- A polynomial-time algorithm for any one of them would imply there exist polynomial-time algorithms for all of them

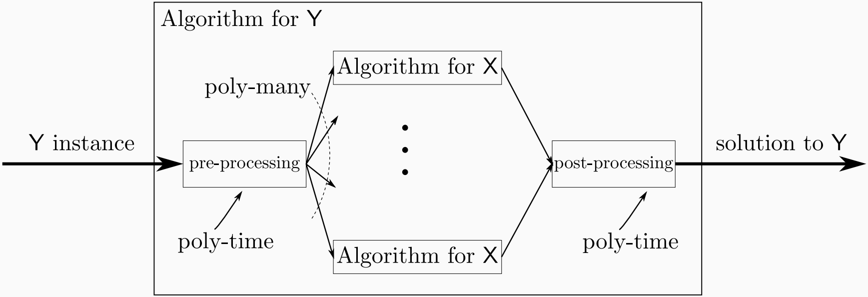
Tool: **polynomial-time** reduction

Polynomial-time reduction

Definition

A problem Y is **polynomial-time reducible** to a problem X if there exists an algorithm that solves any instance of Y making polynomially many elementary operations and polynomially many calls to a black-box solving X

Denote it by $\mathsf{Y} \leq_P \mathsf{X}$



Consequences of polynomial time reductions

Lemma

Suppose $\mathsf{Y} \leq_P \mathsf{X}$. *If* X *can be solved in polynomial time, then* Y *can be solved in polynomial time*

Intuition: X is at least as hard as Y

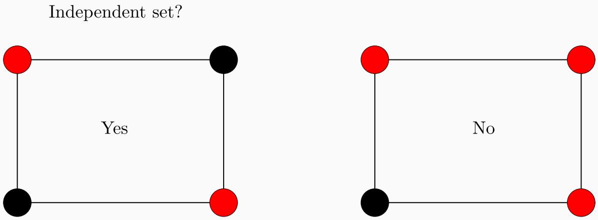
Lemma

Suppose $\mathsf{Y} \leq_P \mathsf{X}$. *If* Y *cannot be solved in polynomial time, then* X *cannot be solved in polynomial time*

Independent set (of a graph)

Definition

A set of vertices is said to be **independent**, if no two of them are connected by an edge



The Maximum Independent Set Problem

The Maximum Independent Set Problem (Decision version)

Instance: a graph G, a number k

Objective: Decide if G contains an independent set of size k?

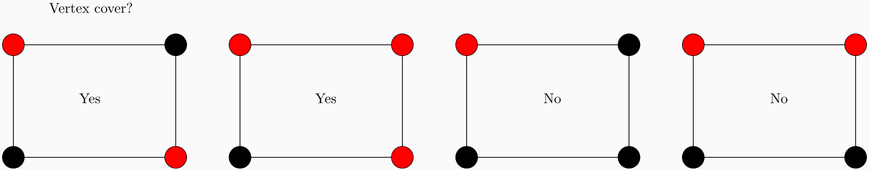
Optimization version: Find the maximum independent set

- Decision version $\leq_P$ optimization version
- Optimization version $\leq_P$ decision version (binary search)

Vertex Cover of a graph

Definition

A set of vertices is said to be a **vertex cover** if every edge has at least one end in it



The Minimum Vertex Cover Problem (Decision version)

Instance: a graph G, a number k

Objective: Decide if G contains a vertex cover of size k?

Independent Set and Vertex Cover (I)

Lemma

Let $G = (V, E)$ be a graph. Then S is an independent set if and only if its complement $V - S$ is a vertex cover

Proof.

- **"only if":** Suppose S is an independent set. Consider an arbitrary edge $e = (u, v)$. We know u, v cannot be both in S — one of them must be in $V - S$. So every edge has at least one end in $V - S$. So $V - S$ is a vertex cover
- **"if":** Suppose $V - S$ is a vertex cover. Consider any two vertices u, v in S. If u, v were joined by an edge, then neither of u, v would be in $V - S$, contradicting the assumption that $V - S$ is a vertex cover. So no two vertices in S are jointed by an edge. So S is an independent set □

Independent Set vs Vertex Cover (II)

Theorem

- *Independent Set* $\leq_P$ *Vertex Cover*
- *Vertex Cover* $\leq_P$ *Independent Set*