

# *CMSC 451: Reductions & NP-completeness*

Slides By: Carl Kingsford



Department of Computer Science  
University of Maryland, College Park

Based on Section 8.1 of *Algorithm Design* by Kleinberg & Tardos.

# Reductions as tool for hardness

We want prove some problems are computationally difficult.

As a first step, we settle for relative judgements:

Problem  $X$  is at least as hard as problem  $Y$

To prove such a statement, we **reduce** problem  $Y$  to problem  $X$ :

*If you had a black box that can solve instances of problem  $X$ , how can you solve any instance of  $Y$  using polynomial number of steps, plus a polynomial number of calls to the black box that solves  $X$ ?*

# Polynomial Reductions

- If problem  $Y$  can be reduced to problem  $X$ , we denote this by  $Y \leq_P X$ .
- This means “ $Y$  is polynomial-time reducible to  $X$ .”
- It also means that  $X$  is at least as hard as  $Y$  because if you can solve  $X$ , you can solve  $Y$ .
- Note: We reduce *to* the problem we want to show is the harder problem.

# Polynomial Problems

Suppose:

- $Y \leq_P X$ , and
- there is an polynomial time algorithm for  $X$ .

Then, there is a polynomial time algorithm for  $Y$ .

Why?

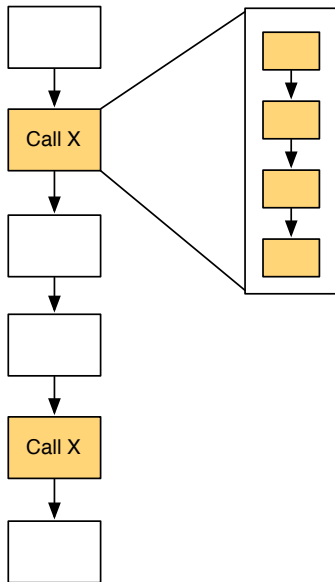
# Polynomial Problems

Suppose:

- $Y \leq_P X$ , and
- there is a polynomial time algorithm for  $X$ .

Then, there is a polynomial time algorithm for  $Y$ .

**Why?** Because polynomials compose.



# We've Seen Reductions Before

Examples of Reductions:

- MAX BIPARTITE MATCHING  $\leq_P$  MAX NETWORK FLOW.
- IMAGE SEGMENTATION  $\leq_P$  MIN-CUT.
- SURVEY DESIGN  $\leq_P$  MAX NETWORK FLOW.
- DISJOINT PATHS  $\leq_P$  MAX NETWORK FLOW.

# Reductions for Hardness

## Theorem

*If  $Y \leq_P X$  and  $Y$  cannot be solved in polynomial time, then  $X$  cannot be solved in polynomial time.*

Why? If we *could* solve  $X$  in polynomial time, then we'd be able to solve  $Y$  in polynomial time using the reduction, contradicting the assumption.

So: If we could find one hard problem  $Y$ , we could prove that another problem  $X$  is hard by reducing  $Y$  to  $X$ .

# Vertex Cover

**Def.** A **vertex cover** of a graph is a set  $S$  of nodes such that every edge has at least one endpoint in  $S$ .

In other words, we try to “cover” each of the edges by choosing at least one of its vertices.

## Vertex Cover

Given a graph  $G$  and a number  $k$ , does  $G$  contain a vertex cover of size at most  $k$ .



# Independent Set to Vertex Cover

## Independent Set

Given graph  $G$  and a number  $k$ , does  $G$  contain a set of at least  $k$  independent vertices?

Can we reduce independent set to vertex cover?

## Vertex Cover

Given a graph  $G$  and a number  $k$ , does  $G$  contain a vertex cover of size at most  $k$ .

# Relation btw Vertex Cover and Indep. Set

## Theorem

*If  $G = (V, E)$  is a graph, then  $S$  is an independent set  $\iff V - S$  is a vertex cover.*

*Proof.*  $\implies$  Suppose  $S$  is an independent set, and let  $e = (u, v)$  be some edge. Only one of  $u, v$  can be in  $S$ . Hence, at least one of  $u, v$  is in  $V - S$ . So,  $V - S$  is a vertex cover.

$\impliedby$  Suppose  $V - S$  is a vertex cover, and let  $u, v \in S$ . There can't be an edge between  $u$  and  $v$  (otherwise, that edge wouldn't be covered in  $V - S$ ). So,  $S$  is an independent set.  $\square$

# Independent Set $\leq_P$ Vertex Cover

## Independent Set $\leq_P$ Vertex Cover

To show this, we change any instance of Independent Set into an instance of Vertex Cover:

- Given an instance of Independent Set  $\langle G, k \rangle$ ,
- We ask our Vertex Cover black box if there is a vertex cover  $V - S$  of size  $\leq |V| - k$ .

By our previous theorem,  $S$  is an independent set iff  $V - S$  is a vertex cover. If the Vertex Cover black box said:

*yes*: then  $S$  must be an independent set of size  $\geq k$ .

*no*: then there is no vertex cover  $V - S$  of size  $\leq |V| - k$ , hence there is no independent set of size  $\geq k$ .

# Vertex Cover $\leq_P$ Independent Set

Actually, we also have:

Vertex Cover  $\leq_P$  Independent Set

*Proof.* To decide if  $G$  has a vertex cover of size  $k$ , we ask if it has an independent set of size  $n - k$ .  $\square$

So: VERTEX COVER and INDEPENDENT SET are equivalently difficult.

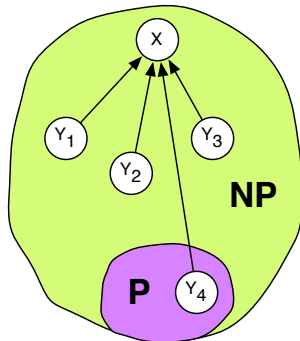
# NP-completeness

**Def.** We say  $X$  is **NP-complete** if:

- $X \in \mathbf{NP}$
- for all  $Y \in \mathbf{NP}$ ,  $Y \leq_P X$ .

If these hold, then  $X$  can be used to solve every problem in **NP**.

Therefore,  $X$  is definitely at least as hard as every problem in **NP**.



# NP-completeness and $P=NP$

## Theorem

*If  $X$  is NP-complete, then  $X$  is solvable in polynomial time if and only if  $P = NP$ .*

*Proof.* If  $P = NP$ , then  $X$  can be solved in polytime.

Suppose  $X$  is solvable in polytime, and let  $Y$  be any problem in  $NP$ . We can solve  $Y$  in polynomial time: reduce it to  $X$ .

Therefore, every problem in  $NP$  has a polytime algorithm and  $P = NP$ .

# Reductions and NP-completeness

## Theorem

*If  $Y$  is NP-complete, and*

①  *$X$  is in NP*

②  *$Y \leq_P X$*

*then  $X$  is NP-complete.*

In other words, we can prove a new problem is NP-complete by reducing some other NP-complete problem to it.

*Proof.* Let  $Z$  be any problem in **NP**. Since  $Y$  is NP-complete,  $Z \leq_P Y$ . By assumption,  $Y \leq_P X$ . Therefore:  $Z \leq_P Y \leq_P X$ .  $\square$

# Some First NP-complete problem

We need to find some first NP-complete problem.

Finding the first NP-complete problem was the result of the Cook-Levin theorem.

We'll deal with this later. For now, trust me that:

- Independent Set is a *packing problem* and is NP-complete.
- Vertex Cover is a *covering problem* and is NP-complete.



# Set Cover

Another very general and useful covering problem:

## Set Cover

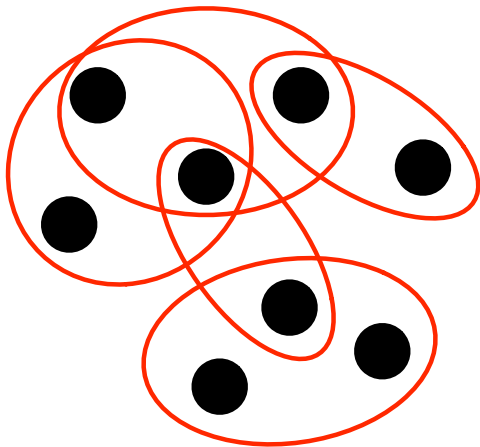
Given a set  $U$  of elements and a collection  $S_1, \dots, S_m$  of subsets of  $U$ , is there a collection of at most  $k$  of these sets whose union equals  $U$ ?

We will show that

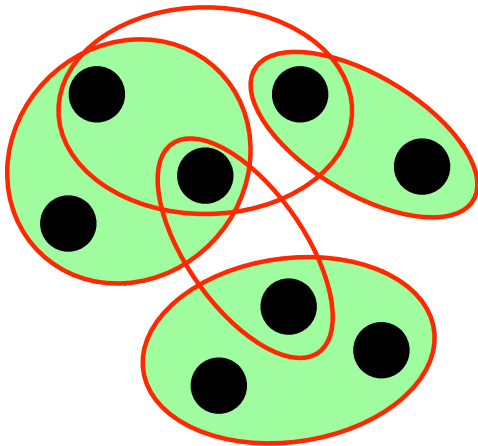
$$\begin{aligned} \text{SET COVER} &\in NP \\ \text{VERTEX COVER} &\leq_P \text{SET COVER} \end{aligned}$$

And therefore that SET COVER is NP-complete.

## Set Cover, Figure



## Set Cover, Figure



# Vertex Cover $\leq_P$ Set Cover

**Thm.** Vertex Cover  $\leq_P$  Set Cover

*Proof.* Let  $G = (V, E)$  and  $k$  be an instance of VERTEX COVER.  
Create an instance of SET COVER:

- $U = E$
- Create a  $S_u$  for each  $u \in V$ , where  $S_u$  contains the edges adjacent to  $u$ .

$U$  can be covered by  $\leq k$  sets iff  $G$  has a vertex cover of size  $\leq k$ .

Why? If  $k$  sets  $S_{u_1}, \dots, S_{u_k}$  cover  $U$  then every edge is adjacent to at least one of the vertices  $u_1, \dots, u_k$ , yielding a vertex cover of size  $k$ .

If  $u_1, \dots, u_k$  is a vertex cover, then sets  $S_{u_1}, \dots, S_{u_k}$  cover  $U$ .  $\square$

## Last Step:

We still have to show that Set Cover is in **NP**!

The certificate is a list of  $k$  sets from the given collection.

We can check in polytime whether they cover all of  $U$ .

Since we have a certificate that can be checked in polynomial time, Set Cover is in **NP**.

# Summary

You can prove a problem is NP-complete by reducing a known NP-complete problem to it.

We know the following problems are NP-complete:

- Vertex Cover
- Independent Set
- Set Cover

Warning: You should reduce the *known* NP-complete problem to the problem you are interested in. (You *will* mistakenly do this backwards sometimes.)