

# Week 9: NP and Computational Intractability

CS 180: Discussion Section 1F

Noor Nakhaei - May 28th, 2021

P vs. NP

# P

- **Decision Problem:**

- Problem  $X$  is a set of strings.
- Instance  $s$  is one string.

- Algorithm  $A$  solves problem  $X$ :  $A(s) : \begin{cases} \text{yes} & \text{if } s \in X \\ \text{no} & \text{if } s \notin X \end{cases}$

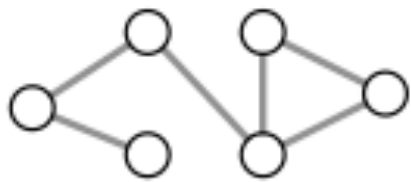
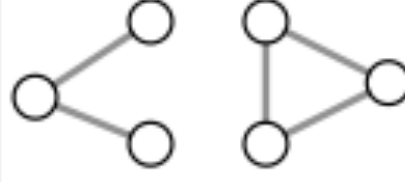
- **Definition:** Algorithm  $A$  runs in **polynomial time** if for every string  $s$ ,  $A(s)$  terminates in  $\leq p(|s|)$  “steps,” where  $p(\cdot)$  is some polynomial function.
- **Definition:**  $P$  = set of decision problems for which there exists a poly-time algorithm.

**problem PRIMES:**  $\{ 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, \dots \}$

**instance  $s$ :** 592335744548702854681

**algorithm:** Agrawal–Kayal–Saxena (2002)

# P

problem	description	poly-time algorithm	yes	no
MULTIPLE	Is $x$ a multiple of $y$ ?	grade-school division	51, 17	51, 16
REL-PRIME	Are $x$ and $y$ relatively prime?	Euclid's algorithm	34, 39	34, 51
PRIMES	Is $x$ prime?	Agrawal–Kayal–Saxena	53	51
EDIT-DISTANCE	Is the edit distance between $x$ and $y$ less than 5?	Needleman–Wunsch	niether neither	acgggt ttttta
L-SOLVE	Is there a vector $x$ that satisfies $Ax = b$ ?	Gauss–Edmonds elimination	$\begin{bmatrix} 0 & 1 & 1 \\ 2 & 4 & -2 \\ 0 & 3 & 15 \end{bmatrix}, \begin{bmatrix} 4 \\ 2 \\ 36 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$
U-CONN	Is an undirected graph $G$ connected?	depth-first search		

# NP

- **Definition:** Algorithm  $C(s, t)$  is a **certifier** for problem  $X$  if for every string  $s : s \in X$  iff there exists a string  $t$  such that  $C(s, t) = \text{yes}$ .
- **Definition:** **NP** = set of decision problems for which there exists a poly-time certifier.
  - $C(s, t)$  is a poly-time algorithm.
  - Certificate  $t$  is of polynomial size:  $|t| \leq p(|s|)$  for some polynomial  $p(\cdot)$ .

problem COMPOSITES:	$\{ 4, 6, 8, 9, 10, 12, 14, 15, 16, 18, 20, \dots \}$
instance $s$ :	437669
certificate $t$ :	541 $\longleftarrow 437,669 = 541 \times 809$
certifier $C(s, t)$ :	grade-school division

# Certifiers and certificates: satisfiability

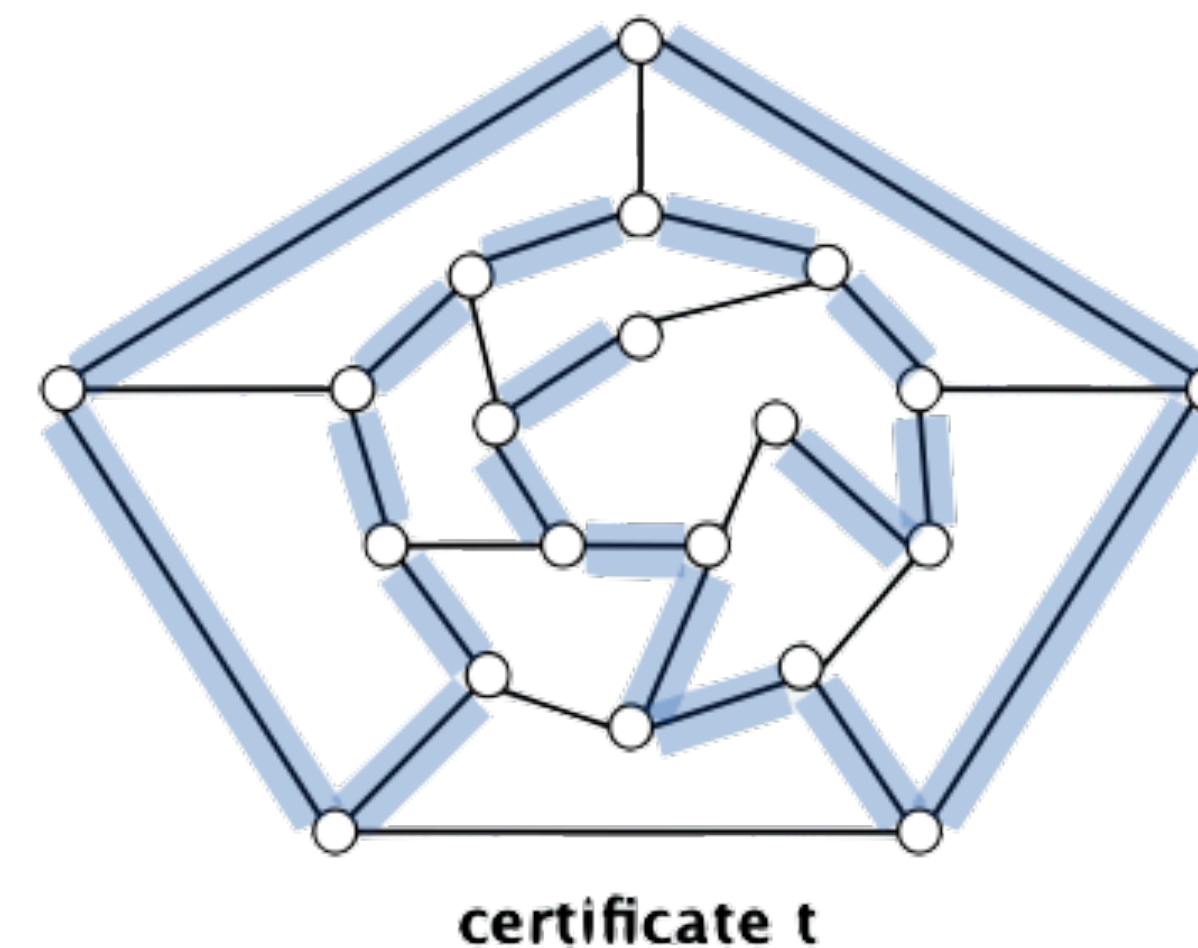
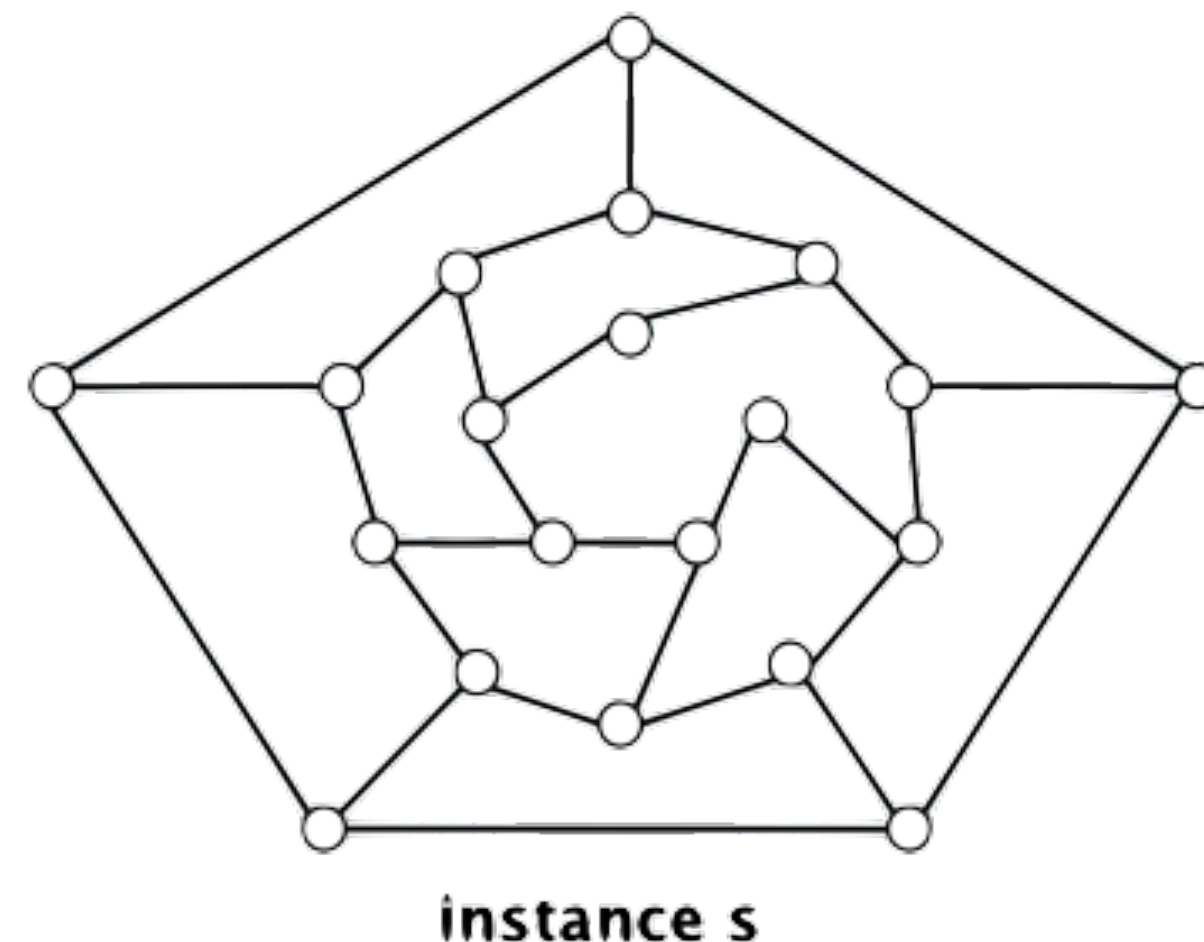
- **SAT:** Given a CNF formula  $\Phi$ , does it have a satisfying truth assignment?
- **3-SAT:** SAT where each clause contains exactly 3 literals.
- **Certificate:** An assignment of truth values to the Boolean variables.
- **Certifier:** Check that each clause in  $\Phi$  has at least one true literal.
- **Conclusion:** SAT  $\in$  NP, 3-SAT  $\in$  NP

$$\text{instance } s \quad \Phi = (\overline{x_1} \vee x_2 \vee x_3) \wedge (x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee x_2 \vee x_4)$$

$$\text{certificate } t \quad x_1 = \text{true}, x_2 = \text{true}, x_3 = \text{false}, x_4 = \text{false}$$

# Certifiers and certificates: Hamilton path

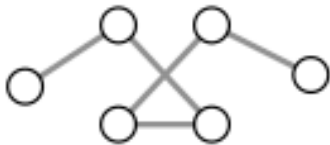
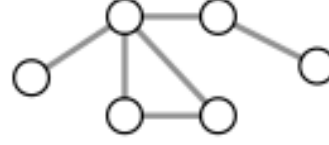
- **Hamilton-Path:** Given an undirected graph  $G = (V, E)$ , does there exist a simple path  $P$  that visits every node?
- **Certificate:** A permutation  $\pi$  of the  $n$  nodes.
- **Certifier:** Check that  $\pi$  contains each node in  $V$  exactly once, and that  $G$  contains an edge between each pair of adjacent nodes.
- **Conclusion:** Hamilton-Path  $\in \mathbf{NP}$





# Some problems in NP

- NP:** Decision problems for which there exists a poly-time certifier.

problem	description	poly-time algorithm	yes	no
L-SOLVE	Is there a vector $x$ that satisfies $Ax = b$ ?	Gauss–Edmonds elimination	$\begin{bmatrix} 0 & 1 & 1 \\ 2 & 4 & -2 \\ 0 & 3 & 15 \end{bmatrix}, \begin{bmatrix} 4 \\ 2 \\ 36 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$
COMPOSITES	Is $x$ composite?	Agrawal–Kayal–Saxena	51	53
FACTOR	Does $x$ have a nontrivial factor less than $y$ ?	???	(56159, 50)	(55687, 50)
SAT	Given a CNF formula, does it have a satisfying truth assignment?	???	$\neg x_1 \vee x_2 \vee \neg x_3$ $x_1 \vee \neg x_2 \vee x_3$ $\neg x_1 \vee \neg x_2 \vee x_3$	$\neg x_2$ $x_1 \vee x_2$ $\neg x_1 \vee x_2$
HAMILTON-PATH	Is there a simple path between $u$ and $v$ that visits every node?	???		



# P, NP, and EXP

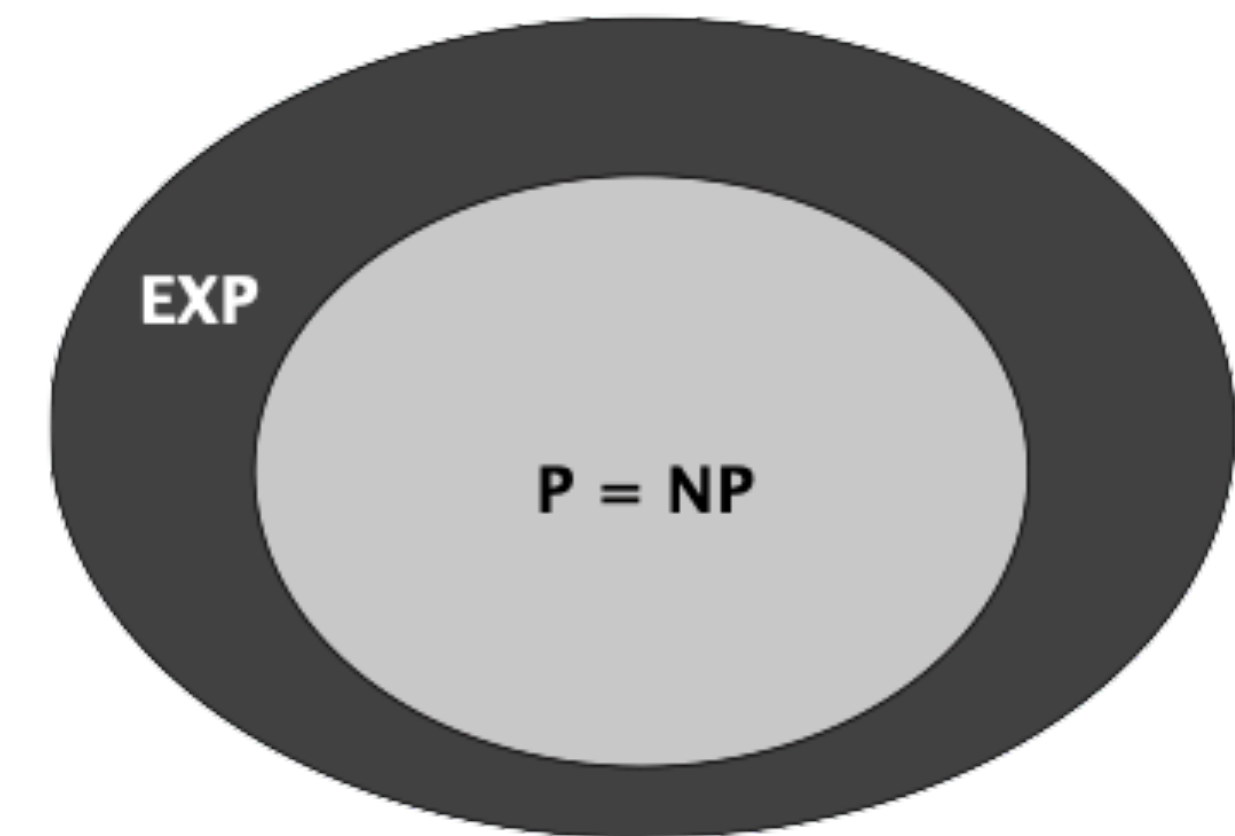
- **P:** Decision problems for which there exists a poly-time algorithm.
- **NP:** Decision problems for which there exists a poly-time certifier.
- **EXP:** Decision problems for which there exists an exponential-time algorithm
- **Proposition:**  $P \subseteq NP$ :
  - **Proof:** Consider any problem  $X \in P$ .
    - By definition, there exists a poly-time algorithm  $A(s)$  that solves  $X$ .
    - Certificate  $t = \varepsilon$ , certifier  $C(s, t) = A(s)$ .
- **Proposition:**  $NP \subseteq EXP$ .
  - **Proof:** Consider any problem  $X \in NP$ .
    - By definition, there exists a poly-time certifier  $C(s, t)$  for  $X$ , where certificate  $t$  satisfies  $|t| \leq p(s)$  for some polynomial  $p(\cdot)$ .
    - To solve instance  $s$ , run  $C(s, t)$  on all strings  $t$  with  $|t| \leq p(|s|)$ .
    - Return yes iff  $C(s, t)$  returns yes for any of these potential certificates.
- **Fact:**  $P \neq EXP \Rightarrow$  either  $P \neq NP$ , or  $NP \neq EXP$ , or both.

# The main question: P vs. NP

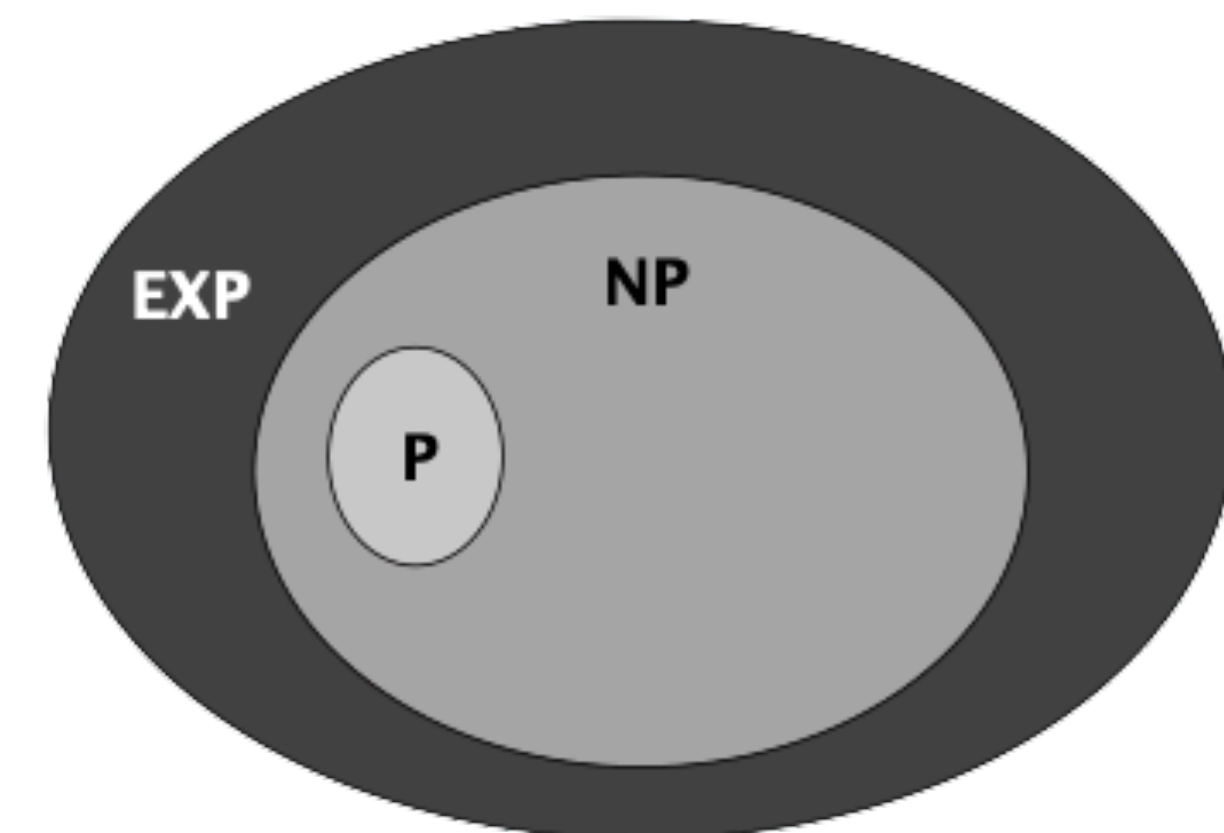
- **Q:** How to solve an instance of 3-SAT with  $n$  variables?
- **A:** Exhaustive search: try all  $2^n$  truth assignments.
- **Q:** Can we do anything substantially more clever?
- **Conjecture:** No poly-time algorithm for 3-SAT.

# The main question: P vs. NP

- **Q:** How to solve an instance of 3-SAT with  $n$  variables?
- **A:** Exhaustive search: try all  $2^n$  truth assignments.
- **Q:** Can we do anything substantially more clever?
- **Conjecture:** No poly-time algorithm for 3-SAT.
- **Does  $P = NP$ ?** Is the decision problem as easy as the certification problem?
- **If yes...** Efficient algorithms for 3-SAT, TSP, Vertex-Cover, Factor, ...
- **If no...** No efficient algorithms possible for 3-SAT, TSP, Vertex-Cover, Factor, ...
- **Consensus opinion.** Probably no.



If  $P = NP$



If  $P \neq NP$

# Possible Outcomes

- $P \neq NP$
- $P = NP$
- $P = NP$ , but only  $\Omega(n^{100})$  algorithm for 3-SAT.
- $P \neq NP$ , but with  $O(n^{\log n})$  algorithm for 3-SAT.
- $P = NP$  is independent (of ZFC axiomatic set theory).

# The main question: P vs. NP

- **NP-Complete:** A problem  $Y \in \mathbf{NP}$  with the property that for every problem  $X \in \mathbf{NP}$ ,  $X \leq_p Y$ .
- **Proposition:** Suppose  $Y \in \mathbf{NP}$ -complete. Then,  $Y \in \mathbf{P}$  iff  $\mathbf{P} = \mathbf{NP}$ .
- **Proof**  $\Leftarrow$  If  $\mathbf{P} = \mathbf{NP}$ , then  $Y \in \mathbf{P}$  because  $Y \in \mathbf{NP}$ .
- **Proof**  $\Leftarrow$  Suppose  $Y \in \mathbf{P}$ .
  - Consider any problem  $X \in \mathbf{NP}$ . Since  $X \leq_p Y$ , we have  $X \in \mathbf{P}$ .
  - This implies  $\mathbf{NP} \subseteq \mathbf{P}$ .
  - We already know  $\mathbf{P} \subseteq \mathbf{NP}$ . Thus  $\mathbf{P} = \mathbf{NP}$ .
- **Fundamental question:** Are there any “natural”  $\mathbf{NP}$ -complete problems?

# Establishing NP-completeness

- **Remark:** Once we establish first “natural” **NP**-complete problem, others fall like dominoes.
- **Recipe:** To prove that  $Y \in \mathbf{NP}$ -complete:
  - Step 1. Show that  $Y \in \mathbf{NP}$ .
  - Step 2. Choose an **NP**-complete problem  $X$ .
  - Step 3. Prove that  $X \leq Y$
- **Proposition:** If  $X \in \mathbf{NP}$ -complete,  $Y \in \mathbf{NP}$ , and  $X \leq_p Y$ , then  $Y \in \mathbf{NP}$ -complete.
- **Proof:** Consider any problem  $W \in \mathbf{NP}$ . Then, both  $W \leq_p X$  and  $X \leq_p Y$ .
  - By transitivity,  $W \leq Y$ .
  - Hence  $Y \in \mathbf{NP}$ -complete.

# Algorithm design patterns and antipatterns

- **Algorithm design patterns:**
  - Greedy
  - Divide and conquer
  - Dynamic programming
  - Duality
  - Reductions
- **Algorithm design antipatterns:**
  - **NP-completeness:**  $O(n^k)$  algorithm unlikely.



# Classify problems according to computational requirements

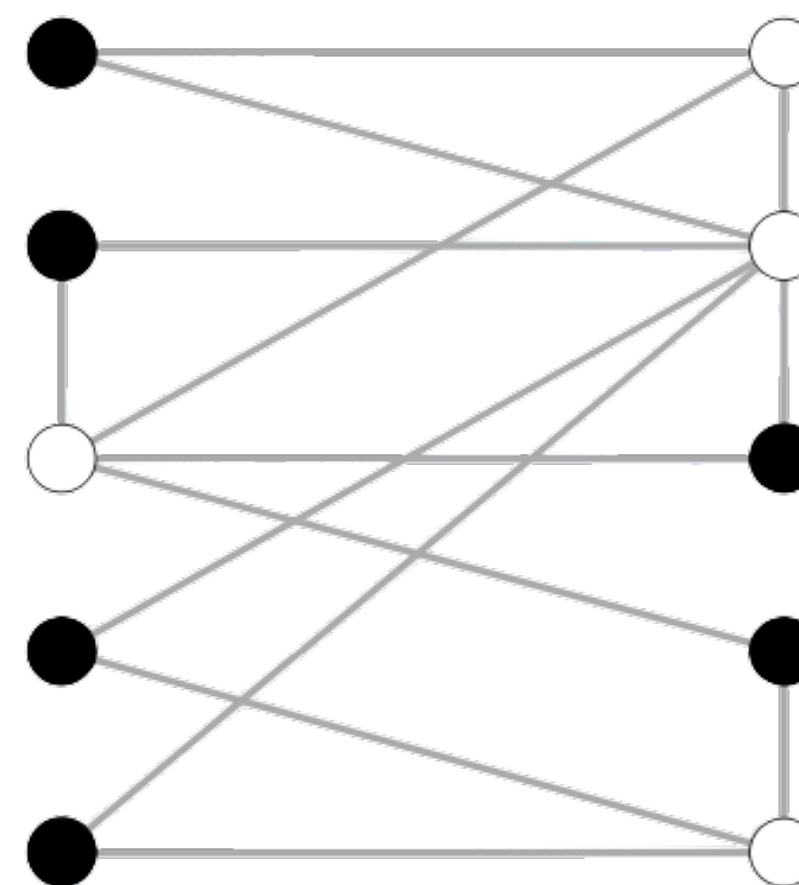
- **Question:** Which problems will we be able to solve in practice?
- **A Working Definition:** Those with poly-time algorithms.

yes	probably no
shortest path	longest path
min cut	max cut
2-satisfiability	3-satisfiability
planar 4-colorability	planar 3-colorability
bipartite vertex cover	vertex cover
matching	3d-matching
primality testing	factoring
linear programming	integer linear programming

# Packing & Covering Problems

# Independent Set

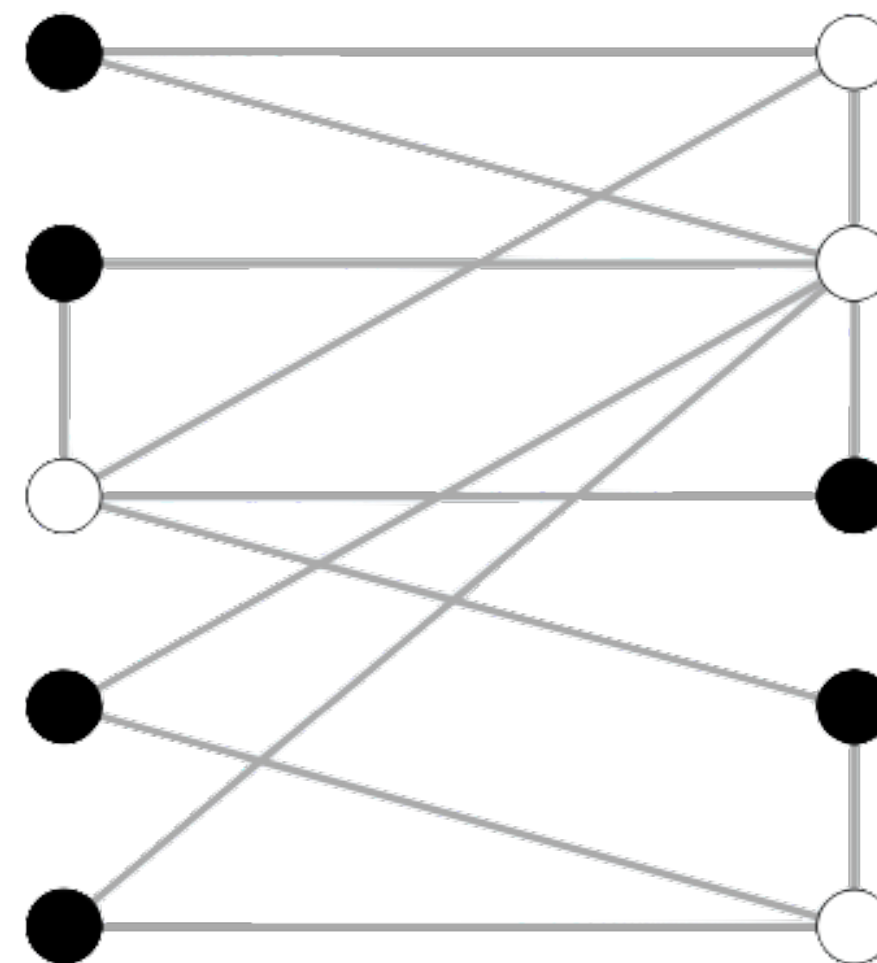
- **Independent-Set:** Given a graph  $G = (V, E)$  and an integer  $k$ , is there a subset of  $k$  (or more) vertices such that no two are adjacent?
- **Example:** Is there an independent set of size  $\geq 6$  ?
- **Example:** Is there an independent set of size  $\geq 7$  ?



● independent set of size 6

# Vertex Cover

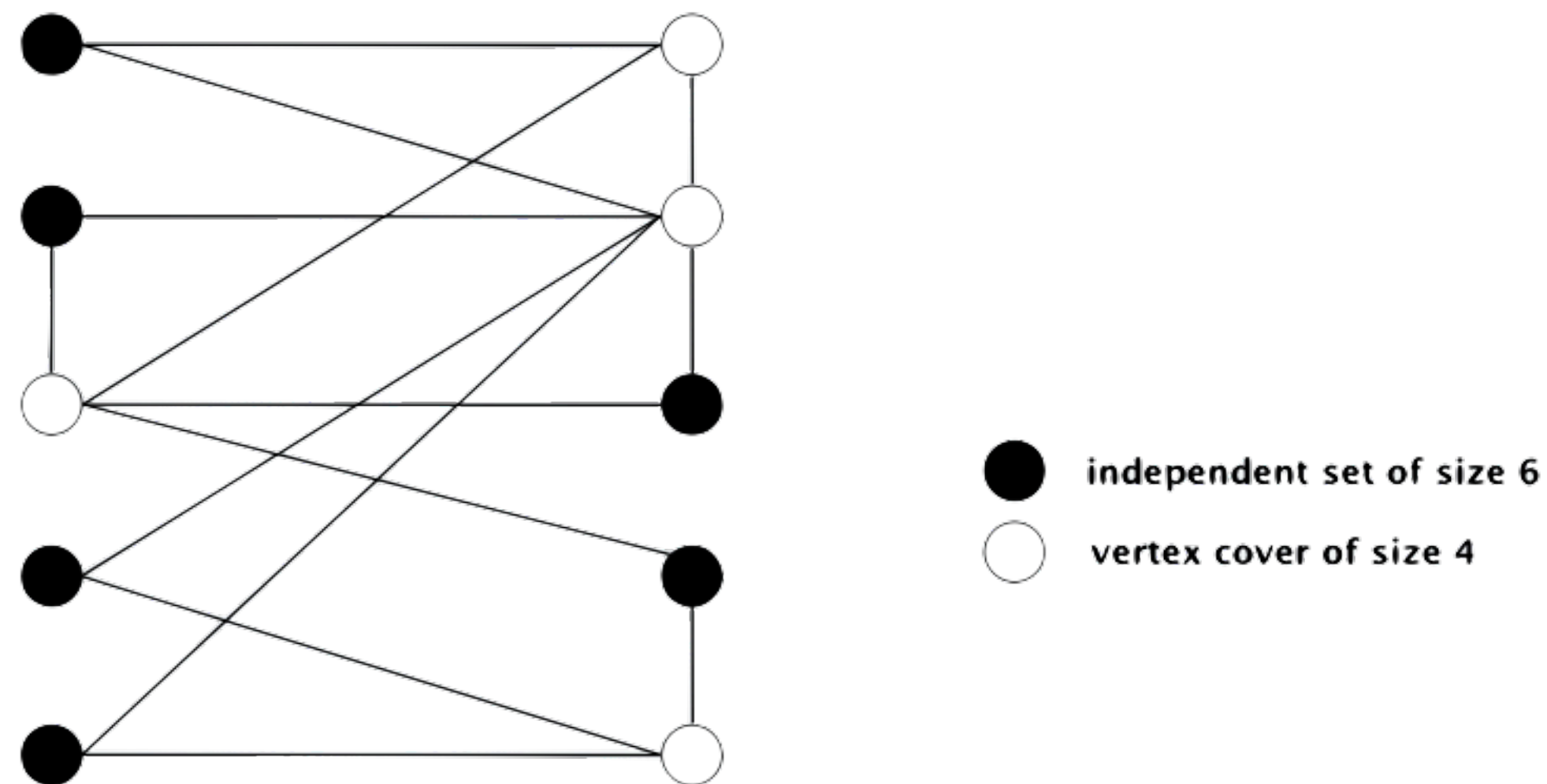
- **Vertex-Cover:** Given a graph  $G = (V, E)$  and an integer  $k$ , is there a subset of  $k$  (or fewer) vertices such that each edge is incident to at least one vertex in the subset?
- **Example:** Is there an vertex cover of size  $\leq 4$  ?
- **Example:** Is there an vertex cover of size  $\leq 3$  ?



● independent set of size 6  
○ vertex cover of size 4

# Vertex cover and independent set reduce to one another

- **Theorem:**  $\text{INDEPENDENT-SET} \equiv_p \text{VERTEX-COVER}$
- **Proof:** We show  $S$  is an independent set of size  $k$  iff  $V - S$  is a vertex cover of size  $n - k$ .



# Vertex cover and independent set reduce to one another

- **Theorem:**  $\text{INDEPENDENT-SET} \equiv_p \text{VERTEX-COVER}$

- **Proof:** We show  $S$  is an independent set of size  $k$  iff  $V - S$  is a vertex cover of size  $n - k$ .

$\Rightarrow$

- Let  $S$  be any independent set of size  $k$ .
- $V - S$  is of size  $n - k$ .
- Consider an arbitrary edge  $(u, v) \in E$ .
- $S$  independent  $\Rightarrow$  either  $u \notin S$ , or  $v \notin S$ , or both, either  $u \in V - S$ , or  $v \in V - S$ , or both.
- Thus,  $V - S$  covers  $(u, v)$ .

# Vertex cover and independent set reduce to one another

- **Theorem:**  $\text{INDEPENDENT-SET} \equiv_p \text{VERTEX-COVER}$

- **Proof:** We show  $S$  is an independent set of size  $k$  iff  $V - S$  is a vertex cover of size  $n - k$ .



- Let  $V-S$  be any independent set of size  $n - k$ .
- $S$  is of size  $k$ .
- Consider an arbitrary edge  $(u, v) \in E$ .
- $V-S$  is a vertex cover  $\Rightarrow$  either  $u \in V - S$ , or  $v \in V - S$ , or both, either  $u \notin S$ , or  $v \notin S$ , or both.
- Thus,  $S$  is an independent set.



# Vertex cover and independent set reduce to one another

- **Set-Cover:** Given a set  $U$  of elements, a collection  $S$  of subsets of  $U$ , and an integer  $k$ , are there  $\leq k$  of these subsets whose union is equal to  $U$  ?
- **Sample application:**
  - $m$  available pieces of software.
  - Set  $U$  of  $n$  capabilities that we would like our system to have.
  - The  $i^{th}$  piece of software provides the set  $S_i \subseteq U$  of capabilities.
  - Goal: achieve all  $n$  capabilities using fewest pieces of software.

$$U = \{ 1, 2, 3, 4, 5, 6, 7 \}$$

$$S_a = \{ 3, 7 \}$$

$$S_b = \{ 2, 4 \}$$

$$S_c = \{ 3, 4, 5, 6 \}$$

$$S_d = \{ 5 \}$$

$$S_e = \{ 1 \}$$

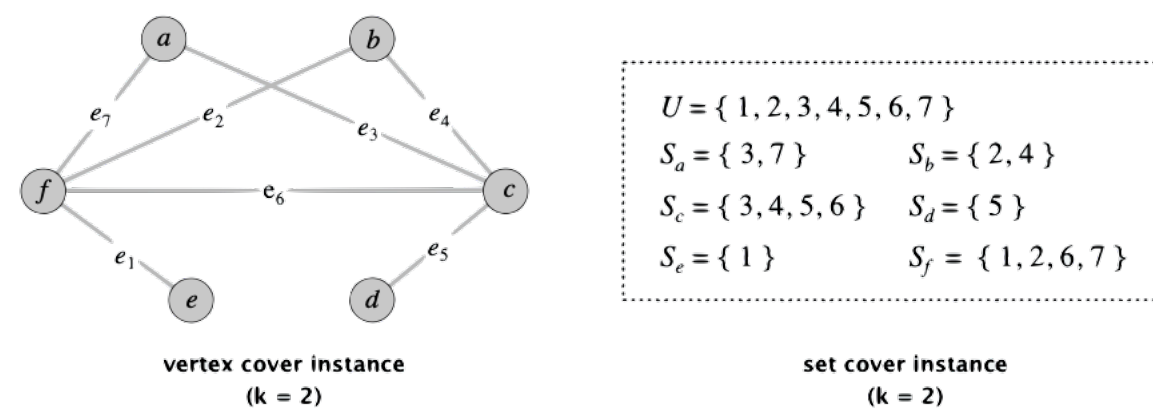
$$S_f = \{ 1, 2, 6, 7 \}$$

$$k = 2$$

a set cover instance

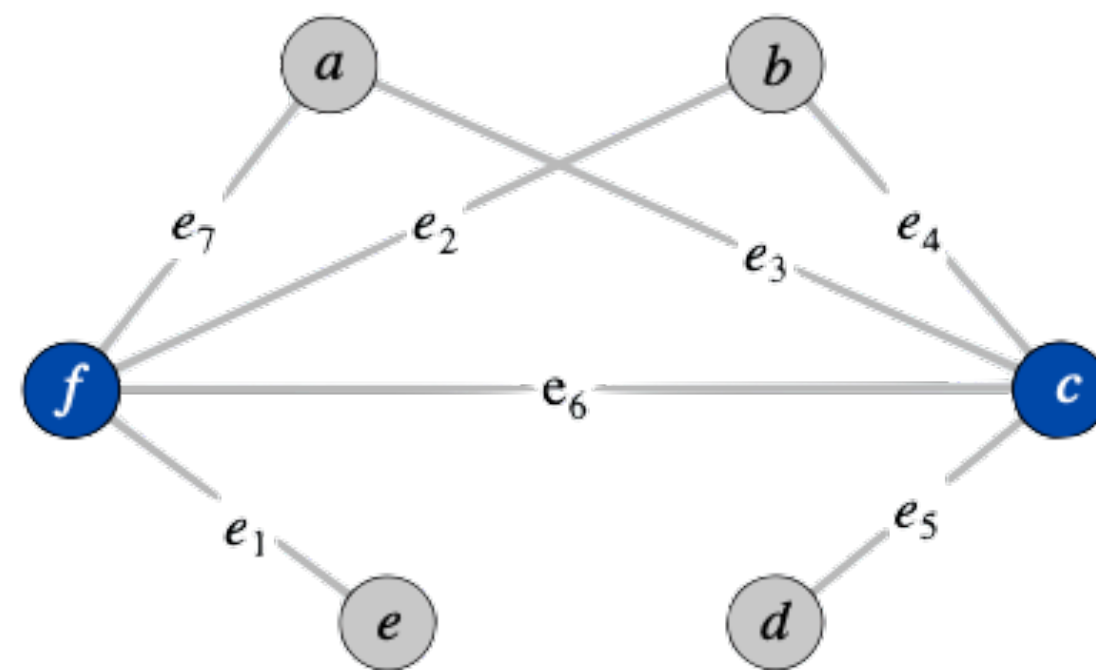
# Vertex cover reduces to set cover

- **Theorem:**  $\text{Vertex-Cover} \leq_p \text{Set-Cover}$
- **Proof:** Given a Vertex-Cover instance  $G = (V, E)$  and  $k$ , we construct a Set-Cover instance  $(U, S, k)$  that has a set cover of size  $k$  iff  $G$  has a vertex cover of size  $k$ .
- **Construction:**
  - Universe  $U = E$ .
  - Include one subset for each node  $v \in V : S_v = \{e \in E : e \text{ incident to } v\}$



# Vertex cover reduces to set cover

- **Lemma:**  $G = (V, E)$  contains a vertex cover of size  $k$  iff  $(U, S, k)$  contains a set cover of size  $k$ .
- **Proof:**  $\Rightarrow$  Let  $X \subseteq V$  be a vertex cover of size  $k$  in  $G$
- Then  $Y = \{ S_v : v \in X \}$  is a set cover of size  $k$ 
  - “yes” instances of Vertex-Cover are solved correctly



vertex cover instance  
( $k = 2$ )

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

set cover instance  
( $k = 2$ )

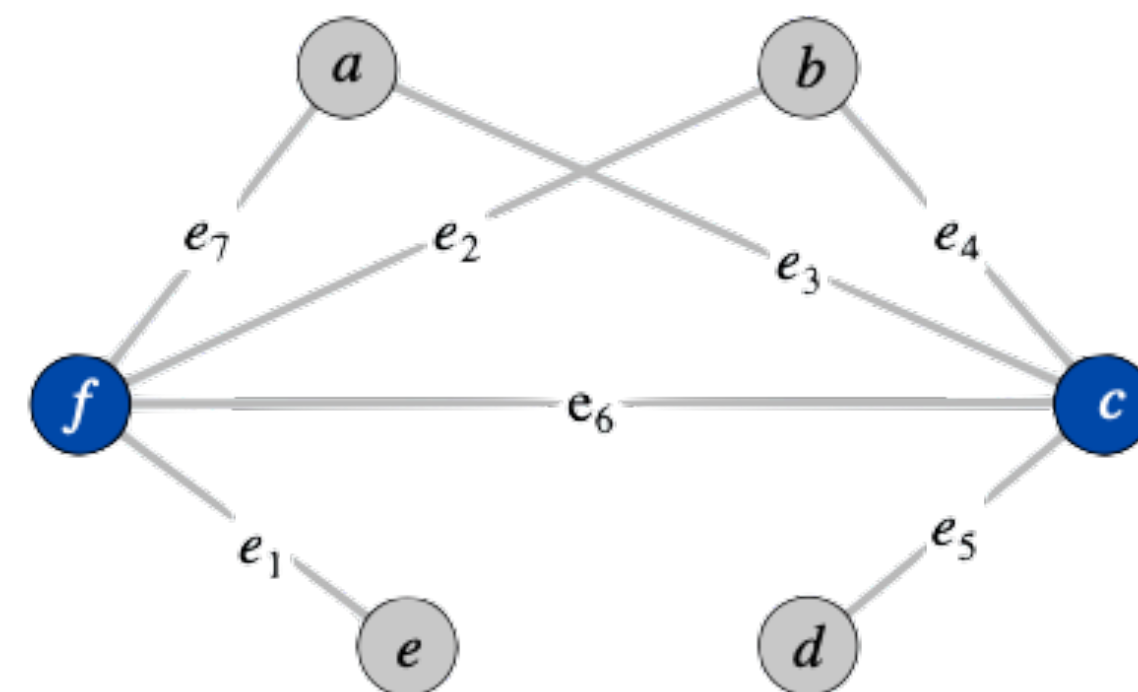
# Vertex cover reduces to set cover

- **Lemma:**  $G = (V, E)$  contains a vertex cover of size  $k$  iff  $(U, S, k)$  contains a set cover of size  $k$ .

- **Proof:**  $\Leftarrow$  Let  $Y \subseteq S$  be a set cover of size  $k$  in  $(U, S, k)$ .

- Then  $X = \{v : S_v \in Y\}$  is a vertex cover of size  $k$  in  $G$ .

- “no” instances of Vertex-Cover are solved correctly



vertex cover instance  
( $k = 2$ )

$U = \{1, 2, 3, 4, 5, 6, 7\}$

$S_a = \{3, 7\}$

$S_b = \{2, 4\}$

$S_c = \{3, 4, 5, 6\}$

$S_d = \{5\}$

$S_e = \{1\}$

$S_f = \{1, 2, 6, 7\}$

set cover instance  
( $k = 2$ )

# Constraint Satisfaction Problems

# Satisfiability

- **Literal:** A Boolean variable or its negation.  $x_i$  or  $\bar{x}_i$
- **Clause:** A disjunction of literals.  $C_j = x_1 \vee \bar{x}_2 \vee x_3$
- **Conjunctive Normal Form (CNF):** A propositional formula  $\Phi$  that is a conjunction of clauses.  $\Phi = C_1 \wedge C_2 \wedge C_3 \wedge C_4$
- **SAT:** Given a CNF formula  $\Phi$ , does it have a satisfying truth assignment?
- **3-SAT:** SAT where each clause contains exactly 3 literals (and each literal corresponds to a different variable).

$$\Phi = (\bar{x}_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \bar{x}_2 \vee x_3) \wedge (\bar{x}_1 \vee x_2 \vee x_4)$$

**yes instance:  $x_1 = \text{true}$ ,  $x_2 = \text{true}$ ,  $x_3 = \text{false}$ ,  $x_4 = \text{false}$**

- **Key-application:** Electronic design automation (EDA).

# Satisfiability

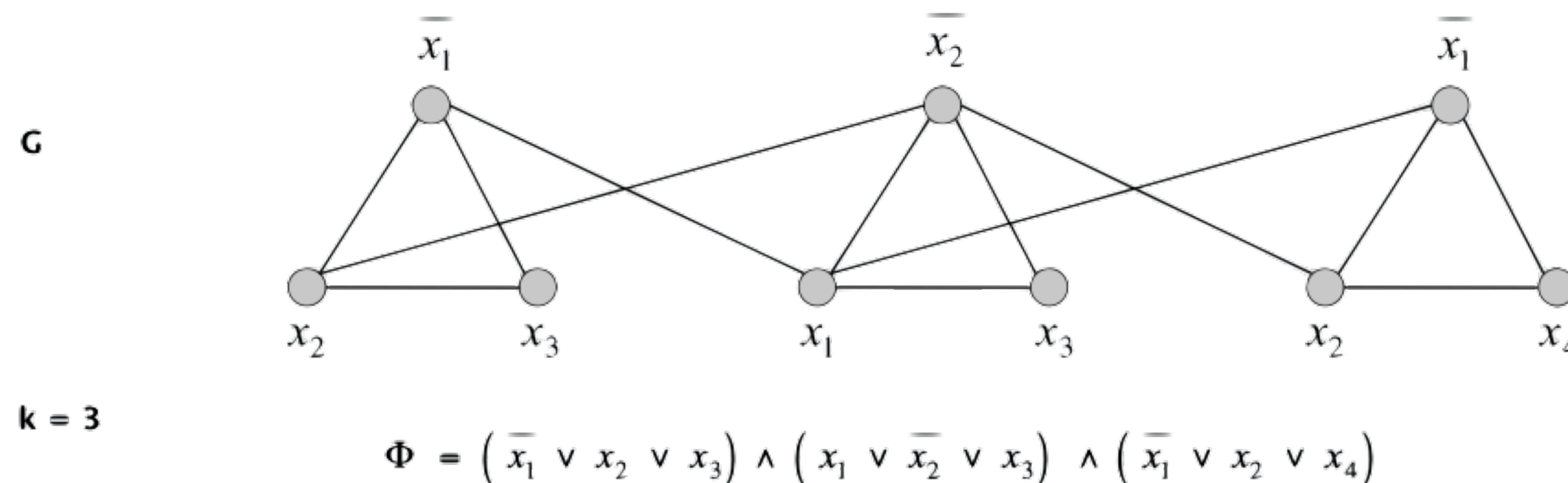
Is hard!

- Scientific Hypothesis: There does not exist a poly-time algorithm for 3-SAT
- $P$  vs.  $NP$ : This hypothesis is equivalent to  $P \neq NP$  conjecture.



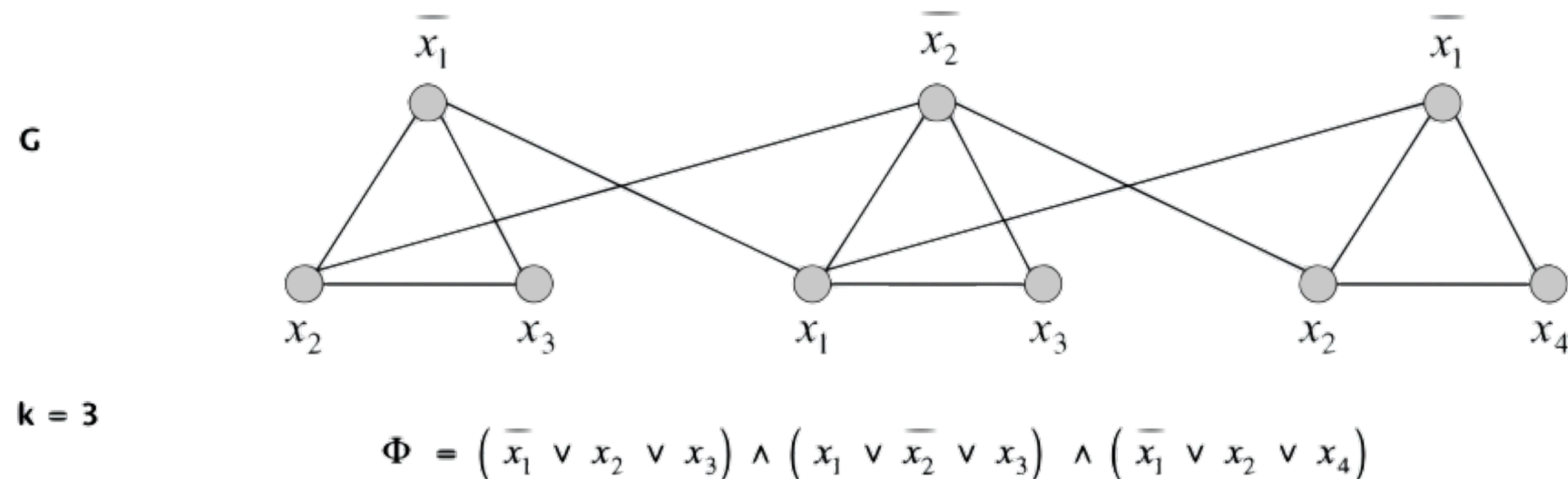
# 3-satisfiability reduces to independent set

- **Theorem:**  $3\text{-SAT} \leq_p \text{Independent-Set}$
- **Proof:** Given an instance  $\Phi$  of 3-SAT, we construct an instance  $(G, k)$  of Independent-Set that has an independent set of size  $k = |\Phi|$  iff  $\Phi$  is satisfiable
- **Construction:**
  - $G$  contains 3 nodes for each clause, one for each literal.
  - Connect 3 literals in a clause in a triangle.
  - Connect literal to each of its negations.



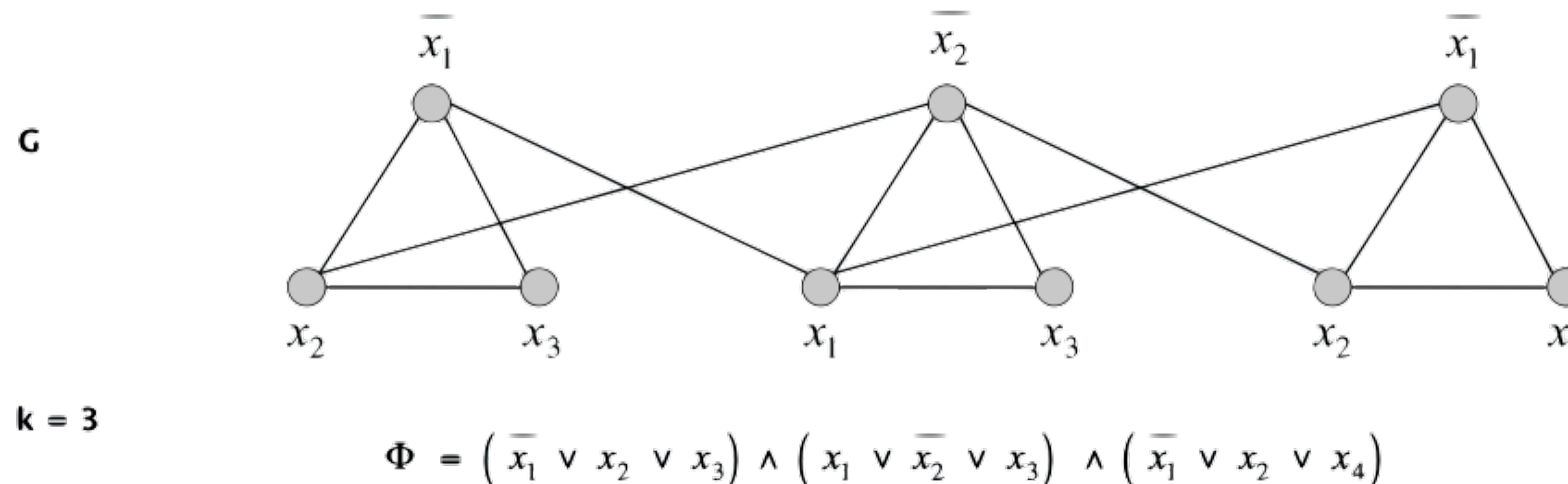
# 3-satisfiability reduces to independent set

- **Lemma:**  $\Phi$  is satisfiable iff  $G$  contains an independent set of size  $k = |\Phi|$
- **Proof:**  $\Rightarrow$ 
  - Consider any satisfying assignment for  $\Phi$ .
  - Select one true literal from each clause/triangle.
  - This is an independent set of size  $k = |\Phi|$ .



# 3-satisfiability reduces to independent set

- **Lemma:**  $\Phi$  is satisfiable iff  $G$  contains an independent set of size  $k = |\Phi|$
- **Proof:**  $\Leftarrow$ 
  - Let  $S$  be independent set of size  $k$ .
  - $S$  must contain exactly one node in each triangle.
  - Set these literals to *true* (and remaining literals consistently).
  - All clauses in  $\Phi$  are satisfied.



# Review

- Basic reduction strategies.
  - Simple equivalence:  $\text{Independent-Set} \equiv_p \text{Vertex-Cover}$
  - Special case to general case:  $\text{Vertex-Cover} \leq_p \text{Set-Cover}$
  - Encoding with gadgets:  $3\text{-SAT} \leq_p \text{Independent-Set}$
- **Transitivity:** If  $X \leq_p Y$  and  $Y \leq_p Z$ , then  $X \leq_p Z$ .
- **Proof idea:** Compose the two algorithms.
- **Example:**  $3\text{-SAT} \leq_p \text{INDEPENDENT-SET} \leq_p \text{VERTEX-COVER} \leq_p \text{SET-COVER}$ .

# Exercises

# Exercise 1

## Question

You're consulting for a small high-tech company that maintains a high-security computer system for some sensitive work that it's doing. To make sure this system is not being used for any illicit purposes, they've set up some logging software that records the IP addresses that all their users are accessing over time. We'll assume that each user accesses at most one IP address in any given minute; the software writes a log file that records, for each user  $u$  and each minute  $m$ , a value  $I(u, m)$  that is equal to the IP address (if any) accessed by user  $u$  during minute  $m$ . It sets  $I(u, m)$  to the null symbol  $\perp$  if  $u$  did not access any IP address during minute  $m$ .

The company management just learned that yesterday the system was used to launch a complex attack on some remote sites. The attack was carried out by accessing  $t$  distinct IP addresses over  $t$  consecutive minutes: In minute 1, the attack accessed address  $i_1$ ; in minute 2, it accessed address  $i_2$ ; and so on, up to address  $i_t$  in minute  $t$ .

Who could have been responsible for carrying out this attack? The company checks the logs and finds to its surprise that there's no single user  $u$  who accessed each of the IP addresses involved at the appropriate time; in other words, there's no  $u$  so that  $I(u, m) = i_m$  for each minute  $m$  from 1 to  $t$ .

So the question becomes: What if there were a small *coalition* of  $k$  users that collectively might have carried out the attack? We will say a subset  $S$  of users is a *suspicious coalition* if, for each minute  $m$  from 1 to  $t$ , there is at least one user  $u \in S$  for which  $I(u, m) = i_m$ . (In other words, each IP address was accessed at the appropriate time by at least one user in the coalition.)

The *Suspicious Coalition Problem* asks: Given the collection of all values  $I(u, m)$ , and a number  $k$ , is there a suspicious coalition of size at most  $k$ ?



# Exercise 1

**Solution** First of all, Suspicious Coalition is clearly in  $\mathcal{NP}$ : If we were to be shown a set  $S$  of users, we could check that  $S$  has size at most  $k$ , and that for each minute  $m$  from 1 to  $t$ , at least one of the users in  $S$  accessed the IP address  $i_m$ .

Now we want to find a known NP-complete problem and reduce it to Suspicious Coalition. Although Suspicious Coalition has lots of features (users, minutes, IP addresses), it's very clearly a covering problem (following the taxonomy described in the chapter): We need to explain all  $t$  suspicious accesses, and we're allowed a limited number of users ( $k$ ) with which to do this. Once we've decided it's a covering problem, it's natural to try reducing Vertex Cover or Set Cover to it. And in order to do this, it's useful to push most of its complicated features into the background, leaving just the bare-bones features that will be used to encode Vertex Cover or Set Cover.

Let's focus on reducing Vertex Cover to it. In Vertex Cover, we need to cover every edge, and we're only allowed  $k$  nodes. In Suspicious Coalition, we need to "cover" all the accesses, and we're only allowed  $k$  users. This parallelism strongly suggests that, given an instance of Vertex Cover consisting of a graph  $G = (V, E)$  and a bound  $k$ , we should construct an instance of Suspicious Coalition in which the users represent the nodes of  $G$  and the suspicious accesses represent the edges.

So suppose the graph  $G$  for the Vertex Cover instance has  $m$  edges  $e_1, \dots, e_m$ , and  $e_j = (v_j, w_j)$ . We construct an instance of Suspicious Coalition as follows. For each node of  $G$  we construct a user, and for each edge  $e_t = (v_t, w_t)$  we construct a minute  $t$ . (So there will be  $m$  minutes total.) In minute  $t$ , the users associated with the two ends of  $e_t$  access an IP address  $i_t$ , and all other users access nothing. Finally, the attack consists of accesses to addresses  $i_1, i_2, \dots, i_m$  in minutes 1, 2,  $\dots$ ,  $m$ , respectively.

The following claim will establish that Vertex Cover  $\leq_P$  Suspicious Coalition and hence will conclude the proof that Suspicious Coalition is NP-complete. Given how closely our construction of the instance shadows the original Vertex Cover instance, the proof is completely straightforward.

**(8.28)** *In the instance constructed, there is a suspicious coalition of size at most  $k$  if and only if the graph  $G$  contains a vertex cover of size at most  $k$ .*

**Proof.** First, suppose that  $G$  contains a vertex cover  $C$  of size at most  $k$ . Then consider the corresponding set  $S$  of users in the instance of Suspicious Coalition. For each  $t$  from 1 to  $m$ , at least one element of  $C$  is an end of the edge  $e_t$ , and the corresponding user in  $S$  accessed the IP address  $i_t$ . Hence the set  $S$  is a suspicious coalition.

Conversely, suppose that there is a suspicious coalition  $S$  of size at most  $k$ , and consider the corresponding set of nodes  $C$  in  $G$ . For each  $t$  from 1 to  $m$ , at least one user in  $S$  accessed the IP address  $i_t$ , and the corresponding node in  $C$  is an end of the edge  $e_t$ . Hence the set  $C$  is a vertex cover. ■



# Exercise 2

## Question

You've been asked to organize a freshman-level seminar that will meet once a week during the next semester. The plan is to have the first portion of the semester consist of a sequence of  $l$  guest lectures by outside speakers, and have the second portion of the semester devoted to a sequence of  $p$  hands-on projects that the students will do.

There are  $n$  options for speakers overall, and in week number  $i$  (for  $i = 1, 2, \dots, l$ ) a subset  $L_i$  of these speakers is available to give a lecture.

On the other hand, each project requires that the students have seen certain background material in order for them to be able to complete the project successfully. In particular, for each project  $j$  (for  $j = 1, 2, \dots, p$ ), there is a subset  $P_j$  of relevant speakers so that the students need to have seen a lecture by *at least one of* the speakers in the set  $P_j$  in order to be able to complete the project.

So this is the problem: Given these sets, can you select exactly one speaker for each of the first  $l$  weeks of the seminar, so that you only choose speakers who are available in their designated week, and so that for each project  $j$ , the students will have seen at least one of the speakers in the relevant set  $P_j$ ? We'll call this the *Lecture Planning Problem*.

To make this clear, let's consider the following sample instance. Suppose that  $l=2$ ,  $p=3$ , and there are  $n=4$  speakers that we denote  $A, B, C, D$ . The availability of the speakers is given by the sets  $L_1 = \{A, B, C\}$  and  $L_2 = \{A, D\}$ . The relevant speakers for each project are given by the sets  $P_1 = \{B, C\}$ ,  $P_2 = \{A, B, D\}$ , and  $P_3 = \{C, D\}$ . Then the answer to this instance of the problem is yes, since we can choose speaker  $B$  in the first week and speaker  $D$  in the second week; this way, for each of the three projects, students will have seen at least one of the relevant speakers.

Prove that Lecture Planning is NP-complete.



# Exercise 2

## Solution

**Solution** The problem is in  $\mathcal{NP}$  since, given a sequence of speakers, we can check (a) all speakers are available in the weeks when they're scheduled, and (b) that for each project, at least one of the relevant speakers has been scheduled.

Now we need to find a known NP-complete problem that we can reduce to Lecture Planning. This is less clear-cut than in the previous exercise, because the statement of the Lecture Planning Problem doesn't immediately map into the taxonomy from the chapter.

There is a useful intuitive view of Lecture Planning, however, that is characteristic of a wide range of constraint satisfaction problems. This intuition is captured, in a strikingly picturesque way, by a description that appeared in the *New Yorker* of the lawyer David Boies's cross-examination style:

*During a cross-examination, David takes a friendly walk down the hall with you while he's quietly closing doors. They get to the end of the hall and David turns on you and there's no place to go. He's closed all the doors.*<sup>3</sup>

What does constraint satisfaction have to do with cross-examination? In Lecture Planning, as in many similar problems, there are two conceptual phases. There's a first phase in which you walk through a set of choices, selecting some and thereby closing the door on others; this is followed by a second phase in which you find out whether your choices have left you with a valid solution or not.

In the case of Lecture Planning, the first phase consists of choosing a speaker for each week, and the second phase consists of verifying that you've picked a relevant speaker for each project. But there are many NP-complete problems that fit this description at a high level, and so viewing Lecture Planning this way helps us search for a plausible reduction. We will in fact describe two reductions, first from 3-SAT and then from Vertex Cover. Of course, either one of these by itself is enough to prove NP-completeness, but both make for useful examples.

3-SAT is the canonical example of a problem with the two-phase structure described above: We first walk through the variables, setting each one to true or false; we then look over each clause and see whether our choices

have satisfied it. This parallel to Lecture Planning already suggests a natural reduction showing that  $3\text{-SAT} \leq_P \text{Lecture Planning}$ : We set things up so that the choice of lecturers sets the variables, and then the feasibility of the projects represents the satisfaction of the clauses.

More concretely, suppose we are given an instance of 3-SAT consisting of clauses  $C_1, \dots, C_k$  over the variables  $x_1, x_2, \dots, x_n$ . We construct an instance of Lecture Planning as follows. For each variable  $x_i$ , we create two lecturers  $z_i$  and  $z'_i$  that will correspond to  $x_i$  and its negation. We begin with  $n$  weeks of lectures; in week  $i$ , the only two lecturers available are  $z_i$  and  $z'_i$ . Then there is a sequence of  $k$  projects; for project  $j$ , the set of relevant lecturers  $P_j$  consists of the three lecturers corresponding to the terms in clause  $C_j$ . Now, if there is a satisfying assignment  $\nu$  for the 3-SAT instance, then in week  $i$  we choose the lecturer among  $z_i, z'_i$  that corresponds to the value assigned to  $x_i$  by  $\nu$ ; in this case, we will select at least one speaker from each relevant set  $P_j$ . Conversely, if we find a way to choose speakers so that there is at least one from each relevant set, then we can set the variables  $x_i$  as follows:  $x_i$  is set to 1 if  $z_i$  is chosen, and it is set to 0 if  $z'_i$  is chosen. In this way, at least one of the three variables in each clause  $C_j$  is set in a way that satisfies it, and so this is a satisfying assignment. This concludes the reduction and its proof of correctness.

Our intuitive view of Lecture Planning leads naturally to a reduction from Vertex Cover as well. (What we describe here could be easily modified to work from Set Cover or 3-Dimensional Matching too.) The point is that we can view Vertex Cover as having a similar two-phase structure: We first choose a set of  $k$  nodes from the input graph, and we then verify for each edge that these choices have covered all the edges.

Given an input to Vertex Cover, consisting of a graph  $G = (V, E)$  and a number  $k$ , we create a lecturer  $z_v$  for each node  $v$ . We set  $\ell = k$ , and define  $L_1 = L_2 = \dots = L_k = \{z_v : v \in V\}$ . In other words, for the first  $k$  weeks, all lecturers are available. After this, we create a project  $j$  for each edge  $e_j = (v, w)$ , with set  $P_j = \{z_v, z_w\}$ .

Now, if there is a vertex cover  $S$  of at most  $k$  nodes, then consider the set of lecturers  $Z_S = \{z_v : v \in S\}$ . For each project  $P_j$ , at least one of the relevant speakers belongs to  $Z_S$ , since  $S$  covers all edges in  $G$ . Moreover, we can schedule all the lecturers in  $Z_S$  during the first  $k$  weeks. Thus it follows that there is a feasible solution to the instance of Lecture Planning.

Conversely, suppose there is a feasible solution to the instance of Lecture Planning, and let  $T$  be the set of all lecturers who speak in the first  $k$  weeks. Let  $X$  be the set of nodes in  $G$  that correspond to lecturers in  $T$ . For each project  $P_j$ , at least one of the two relevant speakers appears in  $T$ , and hence at least one end of each edge  $e_j$  is in the set  $X$ . Thus  $X$  is a vertex cover with at most  $k$  nodes.

This concludes the proof that  $\text{Vertex Cover} \leq_P \text{Lecture Planning}$ .