

## 8. INTRACTABILITY I

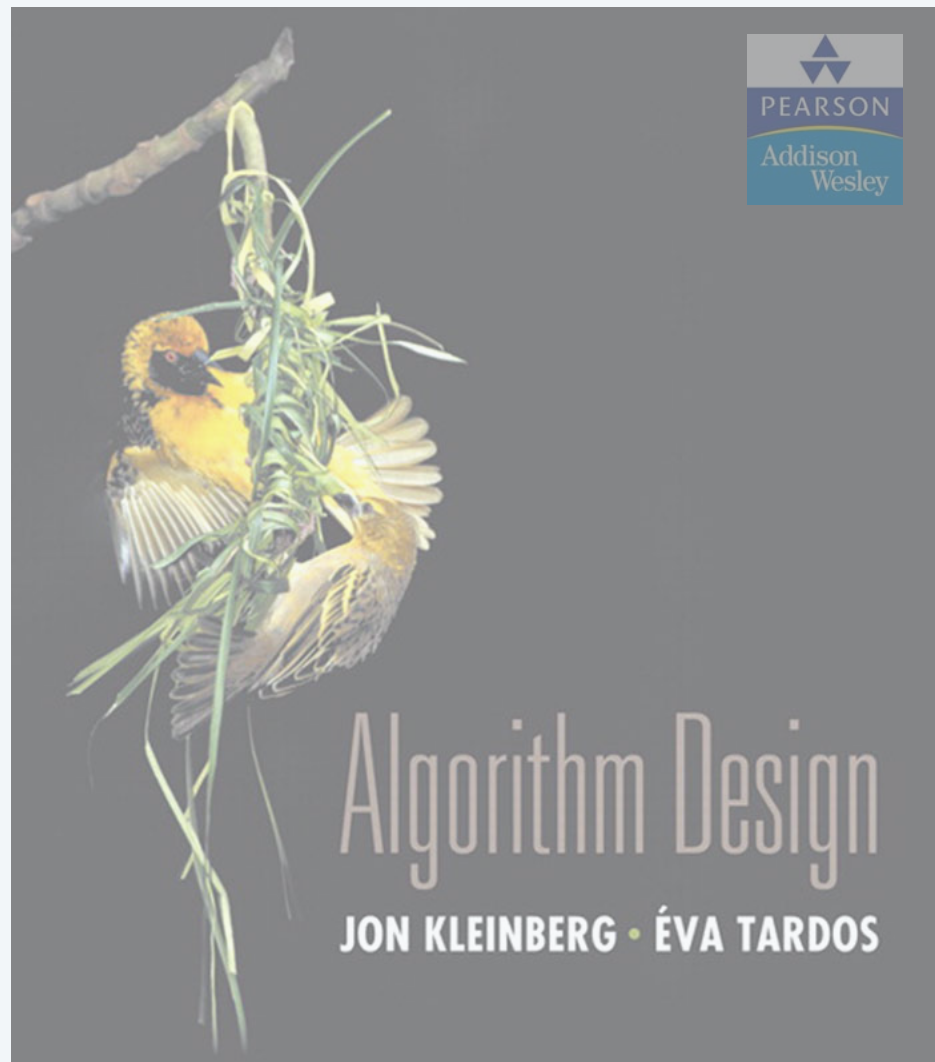
---

- ▶ *poly-time reductions*
- ▶ *packing and covering problems*
- ▶ *constraint satisfaction problems*
- ▶ *sequencing problems*
- ▶ *partitioning problems*
- ▶ *graph coloring*
- ▶ *numerical problems*

Lecture slides by Kevin Wayne

Copyright © 2005 Pearson–Addison Wesley

<http://www.cs.princeton.edu/~wayne/kleinberg-tardos>



## SECTION 8.1

# 8. INTRACTABILITY I

---

- ▶ *poly-time reductions*
- ▶ *packing and covering problems*
- ▶ *constraint satisfaction problems*
- ▶ *sequencing problems*
- ▶ *partitioning problems*
- ▶ *graph coloring*
- ▶ *numerical problems*

# Algorithm design patterns and antipatterns

---

## Algorithm design patterns.

- Greedy.
- Divide and conquer.
- Dynamic programming.
- Duality.
- **Reductions.**
- Local search.
- Randomization.

## Algorithm design antipatterns.

- **NP-completeness.**  $O(n^k)$  algorithm unlikely.
- **PSPACE-completeness.**  $O(n^k)$  certification algorithm unlikely.
- Undecidability. No algorithm possible.

# Classify problems according to computational requirements

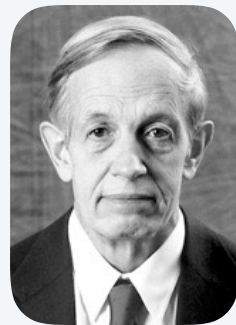
---

Q. Which problems will we be able to solve in practice?

A working definition. Those with poly-time algorithms.



von Neumann  
(1953)



Nash  
(1955)



Gödel  
(1956)



Cobham  
(1964)



Edmonds  
(1965)



Rabin  
(1966)

Turing machine, word RAM, uniform circuits, ...



Theory. Definition is broad and robust.



constants tend to be small, e.g.,  $3n^2$

Practice. Poly-time algorithms scale to huge problems.

# Classify problems according to computational requirements

---

Q. 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

# Classify problems

---

**Desiderata.** Classify problems according to those that can be solved in polynomial time and those that cannot.

**Provably requires exponential time.**

- Given a constant-size program, does it halt in at most  $k$  steps?
- Given a board position in an  $n$ -by- $n$  generalization of checkers, can black guarantee a win?

input size =  $c + \log k$

using forced capture rule



*Alan designed the perfect computer*



**Frustrating news.** Huge number of fundamental problems have defied classification for decades.

# Poly-time reductions

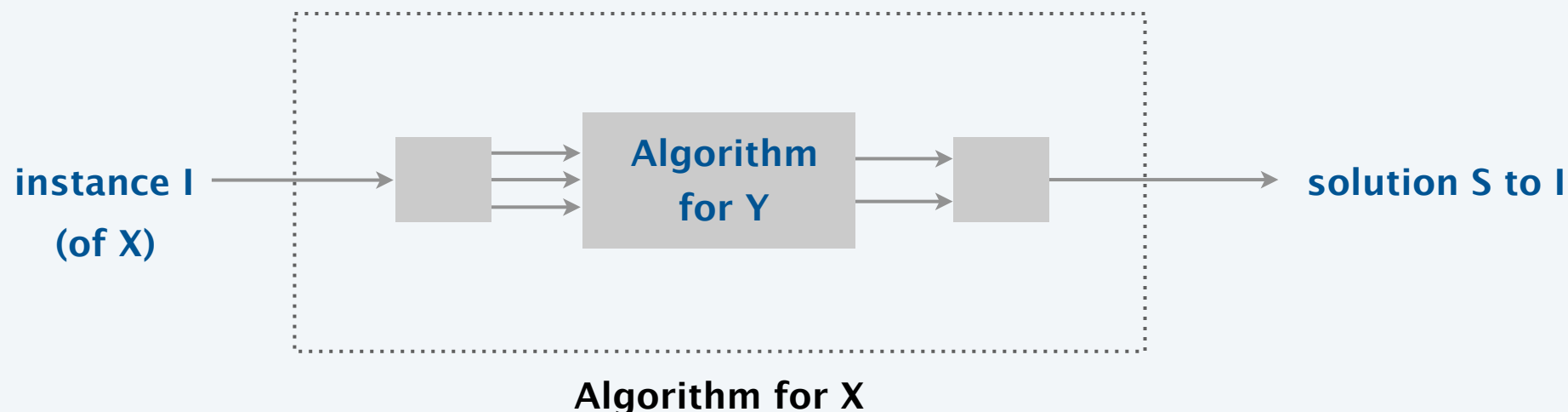
---

**Desiderata'.** Suppose we could solve problem  $Y$  in polynomial time. What else could we solve in polynomial time?

**Reduction.** Problem  $X$  **polynomial-time (Cook) reduces to** problem  $Y$  if arbitrary instances of problem  $X$  can be solved using:

- Polynomial number of standard computational steps, plus
- Polynomial number of calls to oracle that solves problem  $Y$ .

computational model supplemented by special piece of hardware that solves instances of  $Y$  in a single step



# Poly-time reductions

---

**Desiderata'.** Suppose we could solve problem  $Y$  in polynomial time. What else could we solve in polynomial time?

**Reduction.** Problem  $X$  **polynomial-time (Cook) reduces to** problem  $Y$  if arbitrary instances of problem  $X$  can be solved using:

- Polynomial number of standard computational steps, plus
- Polynomial number of calls to oracle that solves problem  $Y$ .

**Notation.**  $X \leq_p Y$ .

**Note.** We pay for time to write down instances of  $Y$  sent to oracle  $\Rightarrow$  instances of  $Y$  must be of polynomial size.

**Novice mistake.** Confusing  $X \leq_p Y$  with  $Y \leq_p X$ .





Suppose that  $X \leq_p Y$ . Which of the following can we infer?

- A.** If  $X$  can be solved in polynomial time, then so can  $Y$ .
- B.**  $X$  can be solved in poly time iff  $Y$  can be solved in poly time.
- C.** If  $X$  cannot be solved in polynomial time, then neither can  $Y$ .
- D.** If  $Y$  cannot be solved in polynomial time, then neither can  $X$ .



Which of the following poly-time reductions are known?

- A.** FIND-MAX-FLOW  $\leq_p$  FIND-MIN-CUT.
- B.** FIND-MIN-CUT  $\leq_p$  FIND-MAX-FLOW.
- C.** Both A and B.
- D.** Neither A nor B.

# Poly-time reductions

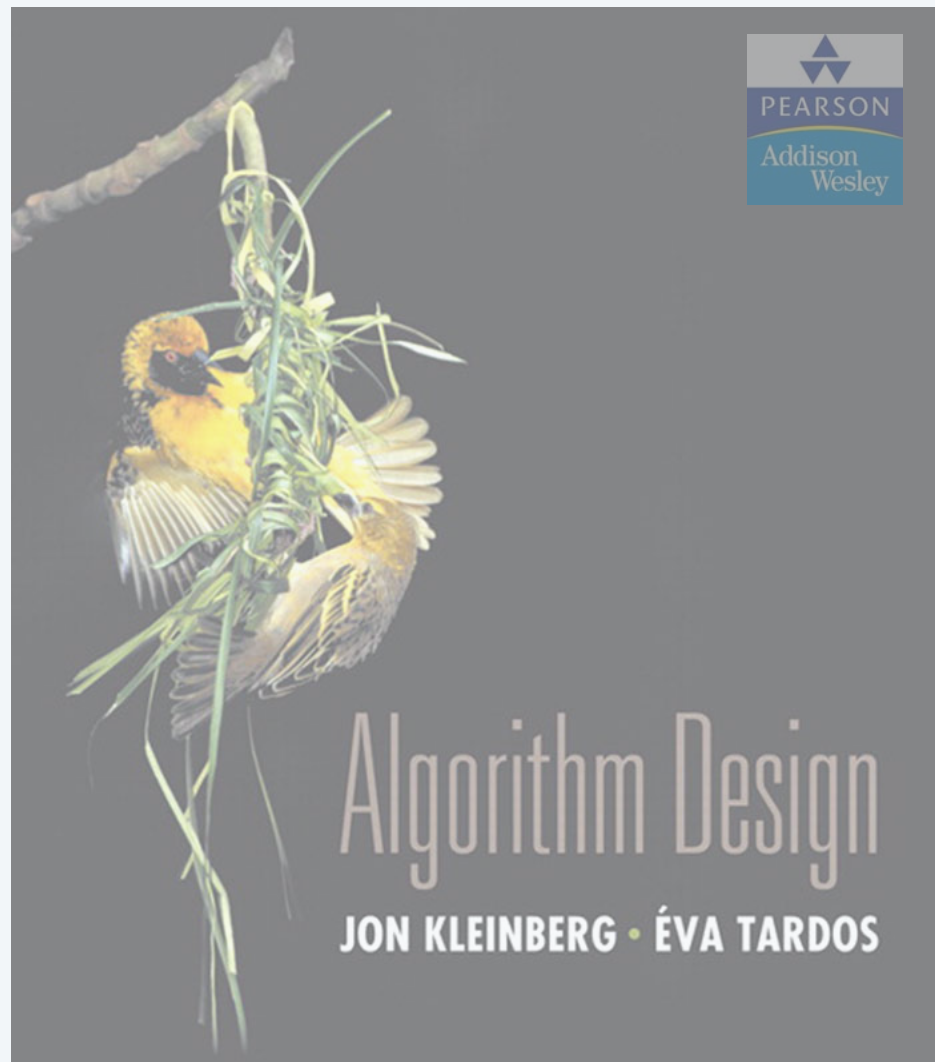
---

**Design algorithms.** If  $X \leq_p Y$  and  $Y$  can be solved in polynomial time, then  $X$  can be solved in polynomial time.

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

**Establish equivalence.** If both  $X \leq_p Y$  and  $Y \leq_p X$ , we use notation  $X \equiv_p Y$ . In this case,  $X$  can be solved in polynomial time iff  $Y$  can be.

**Bottom line.** Reductions classify problems according to **relative** difficulty.



## SECTION 8.1

# 8. INTRACTABILITY I

---

- ▶ *poly-time reductions*
- ▶ *packing and covering problems*
- ▶ *constraint satisfaction problems*
- ▶ *sequencing problems*
- ▶ *partitioning problems*
- ▶ *graph coloring*
- ▶ *numerical problems*

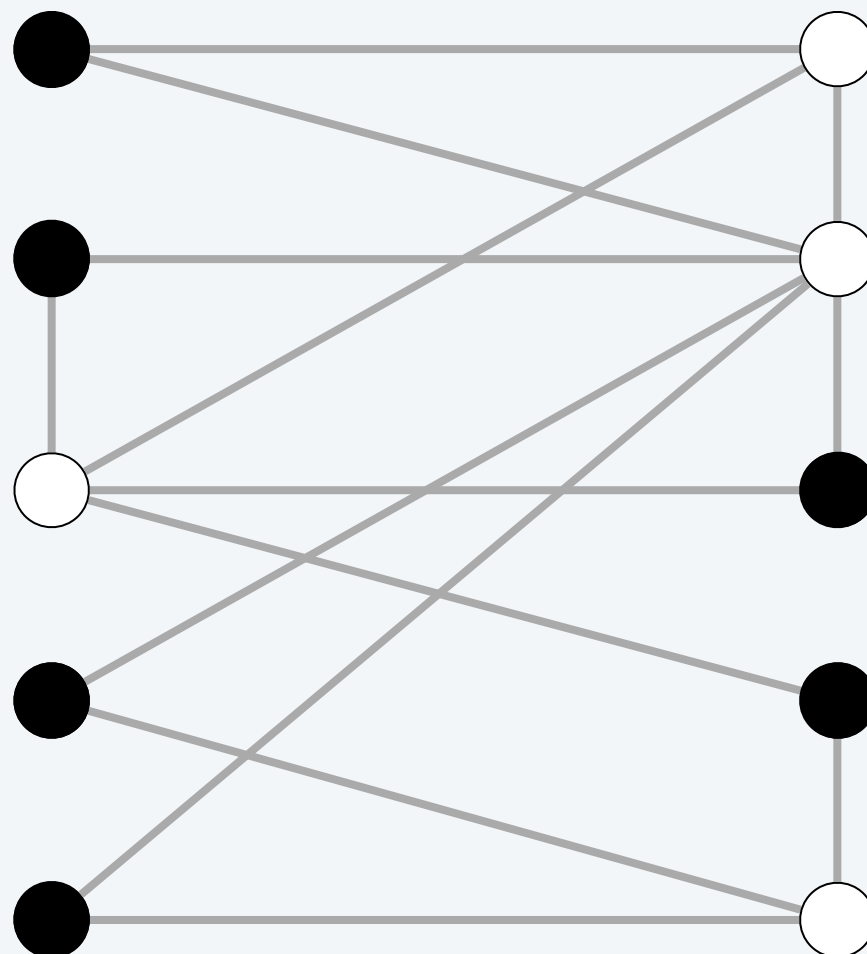
# Independent set

---

**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?

**Ex.** Is there an independent set of size  $\geq 6$ ?

**Ex.** Is there an independent set of size  $\geq 7$ ?



● independent set of size 6

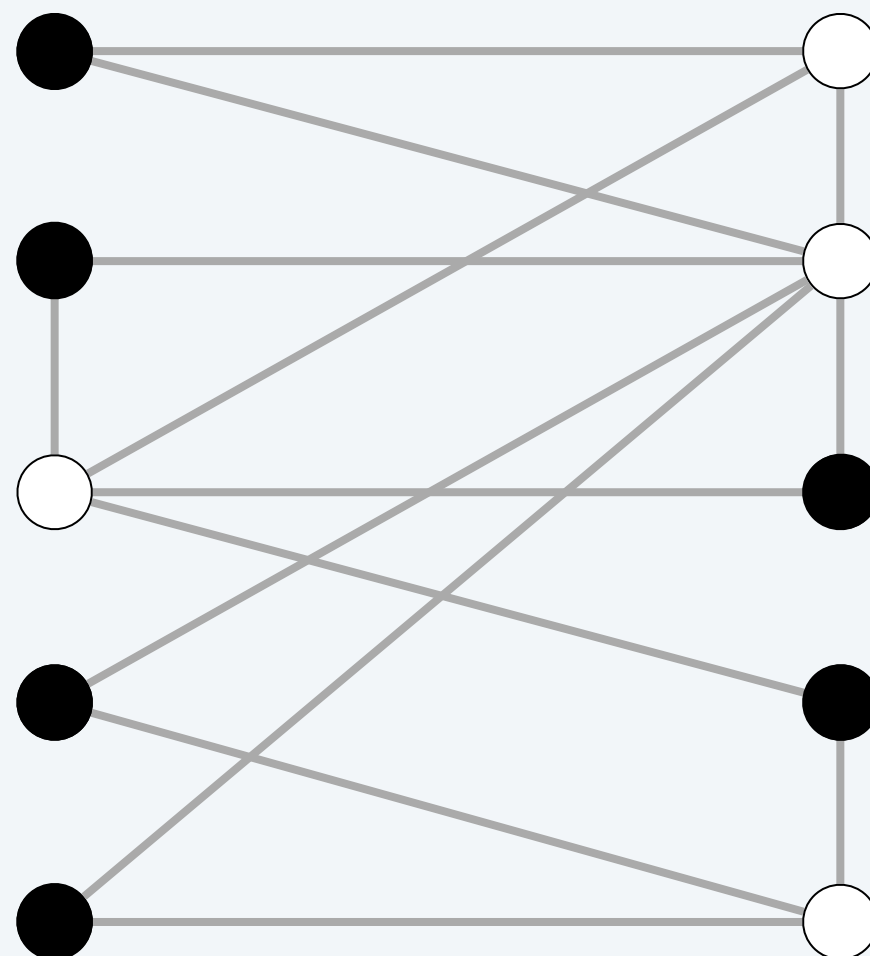
# Vertex cover

---

**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?

**Ex.** Is there a vertex cover of size  $\leq 4$ ?

**Ex.** Is there a vertex cover of size  $\leq 3$ ?

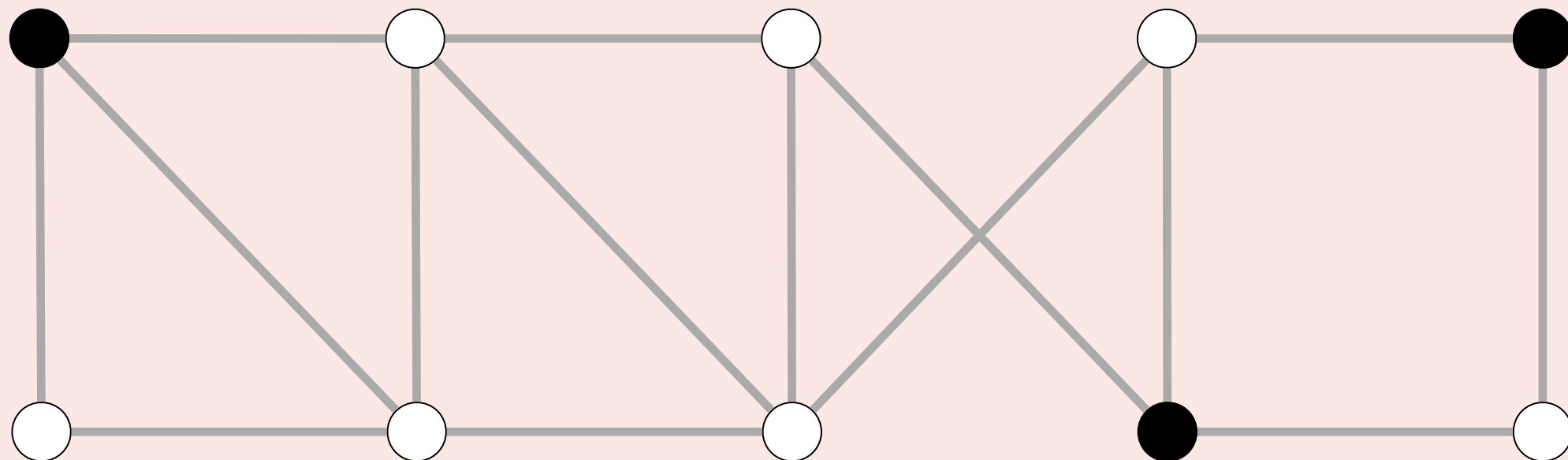


● independent set of size 6  
○ vertex cover of size 4



Consider the following graph  $G$ . Which are true?

- A.** The white vertices are a vertex cover of size 7.
- B.** The black vertices are an independent set of size 3.
- C.** Both A and B.
- D.** Neither A nor B.

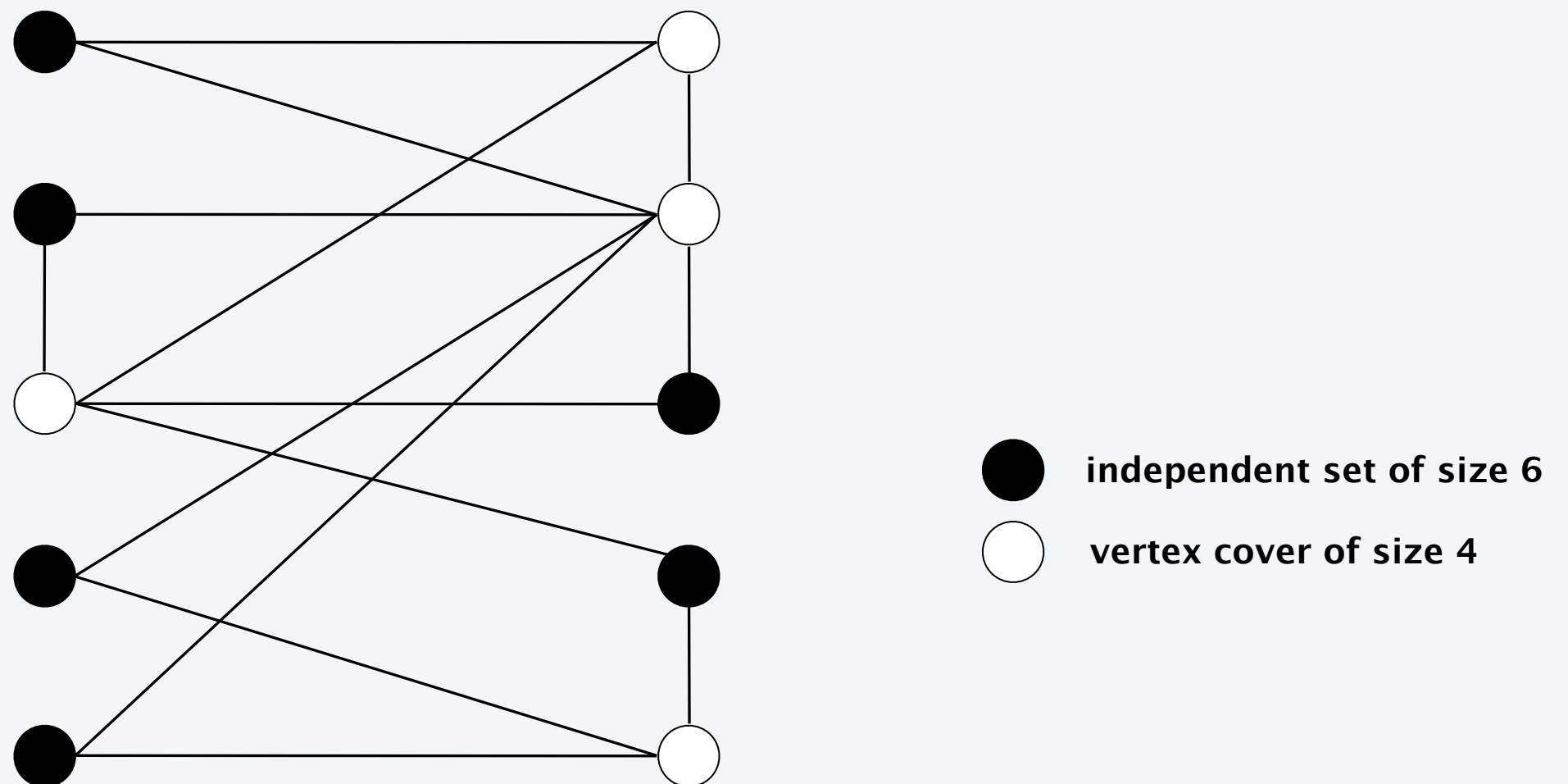


# Vertex cover and independent set reduce to one another

---

**Theorem.**  $\text{INDEPENDENT-SET} \equiv_P \text{VERTEX-COVER}$ .

**Pf.** 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.**  $\text{INDEPENDENT-SET} \equiv_P \text{VERTEX-COVER}$ .

**Pf.** 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.  
 $\Rightarrow$  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.**  $\text{INDEPENDENT-SET} \equiv_P \text{VERTEX-COVER}$ .

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

$\Leftarrow$

- Let  $V - S$  be any vertex cover 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.  
 $\Rightarrow$  either  $u \notin S$ , or  $v \notin S$ , or both.
- Thus,  $S$  is an independent set. ■

# Set cover

---

**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



Given the universe  $U = \{ 1, 2, 3, 4, 5, 6, 7 \}$  and the following sets, which is the minimum size of a set cover?

- A.** 1
- B.** 2
- C.** 3
- D.** None of the above.

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

$$S_a = \{ 1, 4, 6 \}$$

$$S_b = \{ 1, 6, 7 \}$$

$$S_c = \{ 1, 2, 3, 6 \}$$

$$S_d = \{ 1, 3, 5, 7 \}$$

$$S_e = \{ 2, 6, 7 \}$$

$$S_f = \{ 3, 4, 5 \}$$

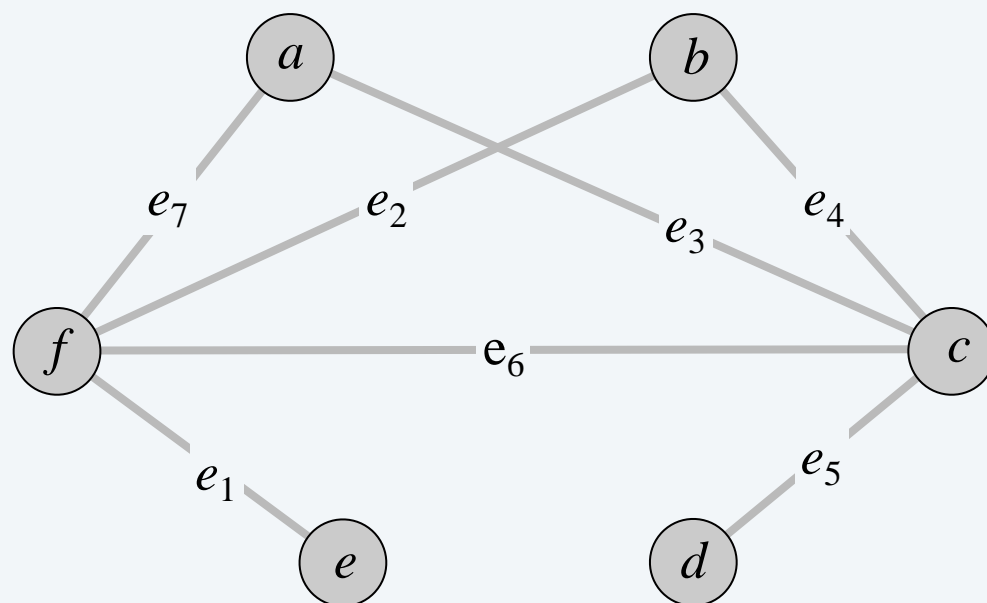
# Vertex cover reduces to set cover

**Theorem.** VERTEX-COVER  $\leq_p$  SET-COVER.

**Pf.** 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 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**  
( $k = 2$ )

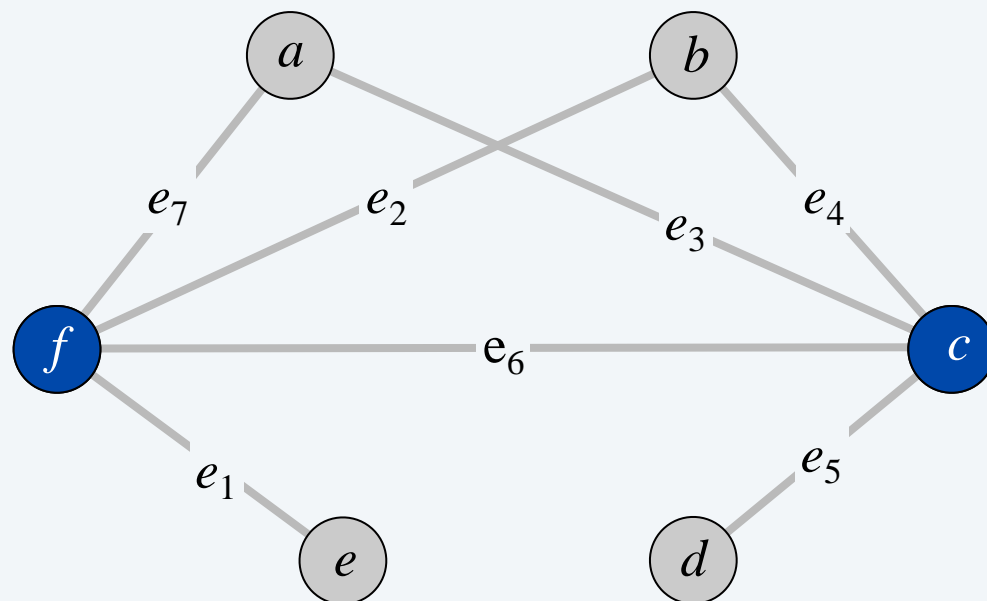
# Vertex cover reduces to set cover

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

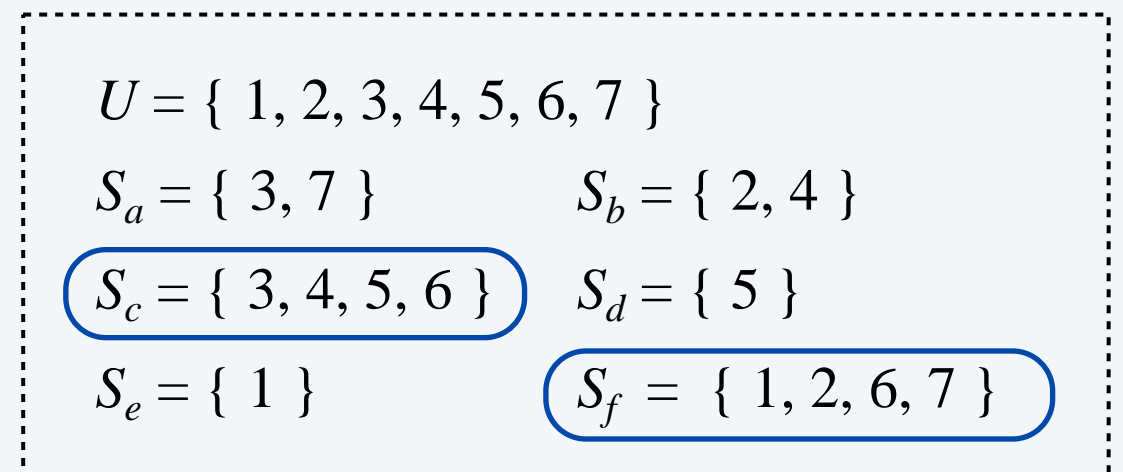
**Pf.**  $\Rightarrow$  Let  $X \subseteq V$  be a vertex cover of size  $k$  in  $G$ .

- 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  
( $k = 2$ )



set cover instance  
( $k = 2$ )

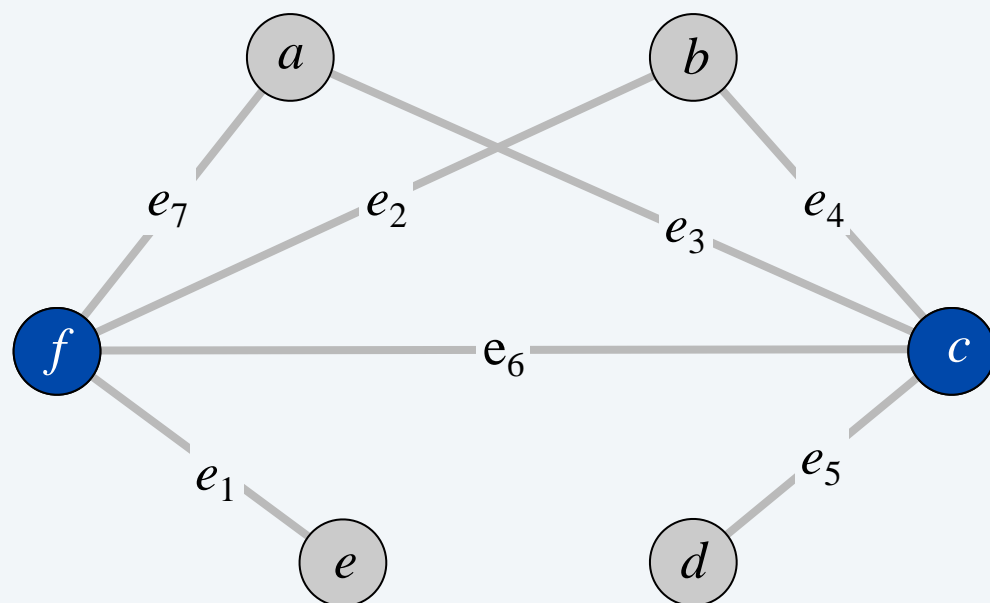
# Vertex cover reduces to set cover

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

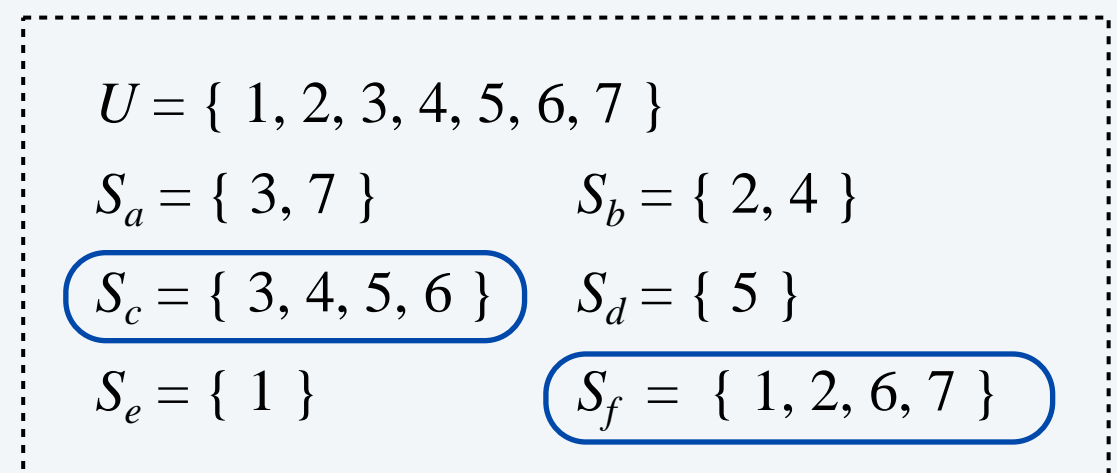
**Pf.**  $\Leftarrow$  Let  $Y \subseteq S$  be a set cover of size  $k$  in  $(U, S, k)$ .

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

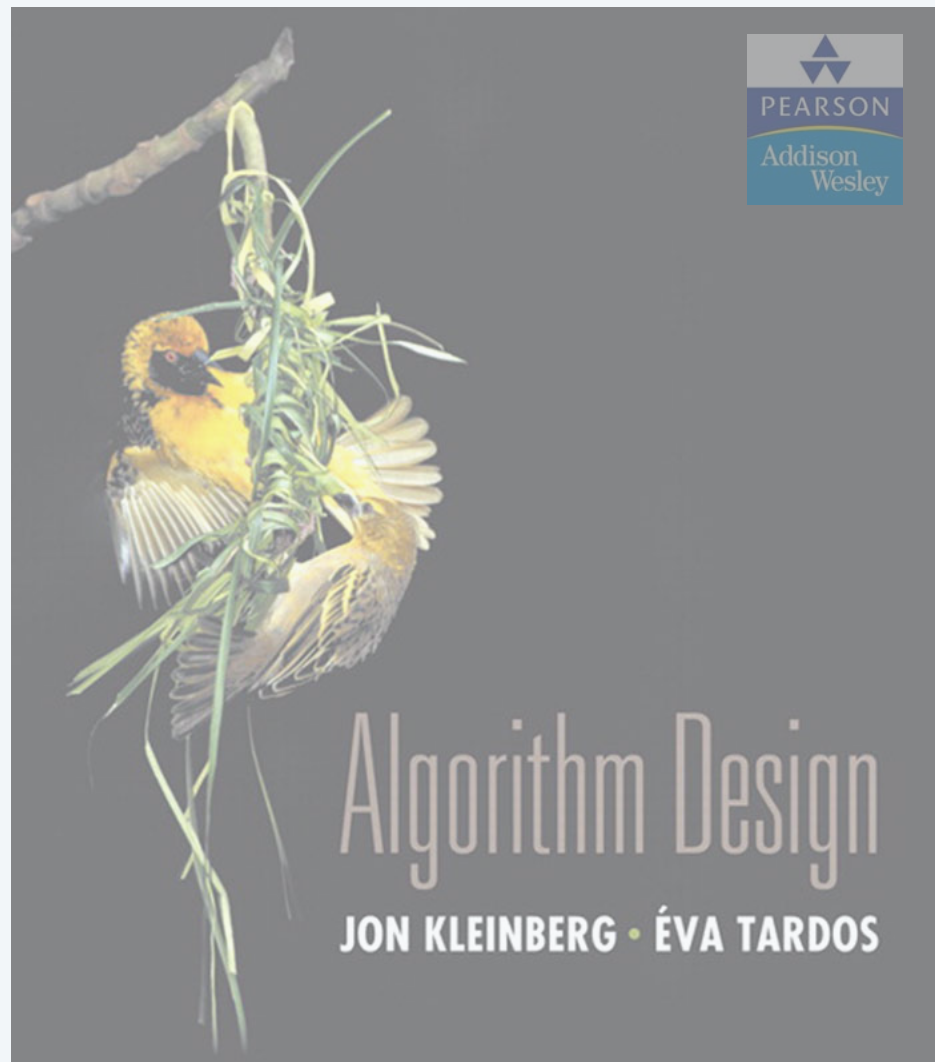
“no” instances of VERTEX-COVER are solved correctly



vertex cover instance  
( $k = 2$ )



set cover instance  
( $k = 2$ )



## SECTION 8.2

# 8. INTRACTABILITY I

---

- ▶ *poly-time reductions*
- ▶ *packing and covering problems*
- ▶ ***constraint satisfaction problems***
- ▶ *sequencing problems*
- ▶ *partitioning problems*
- ▶ *graph coloring*
- ▶ *numerical problems*



# Satisfiability

---

**Literal.** A Boolean variable or its negation.

$$x_i \text{ or } \overline{x_i}$$

**Clause.** A disjunction of literals.

$$C_j = x_1 \vee \overline{x_2} \vee x_3$$

**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).

$$\Phi = (\overline{x_1} \vee x_2 \vee x_3) \wedge (x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee x_2 \vee x_4)$$

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

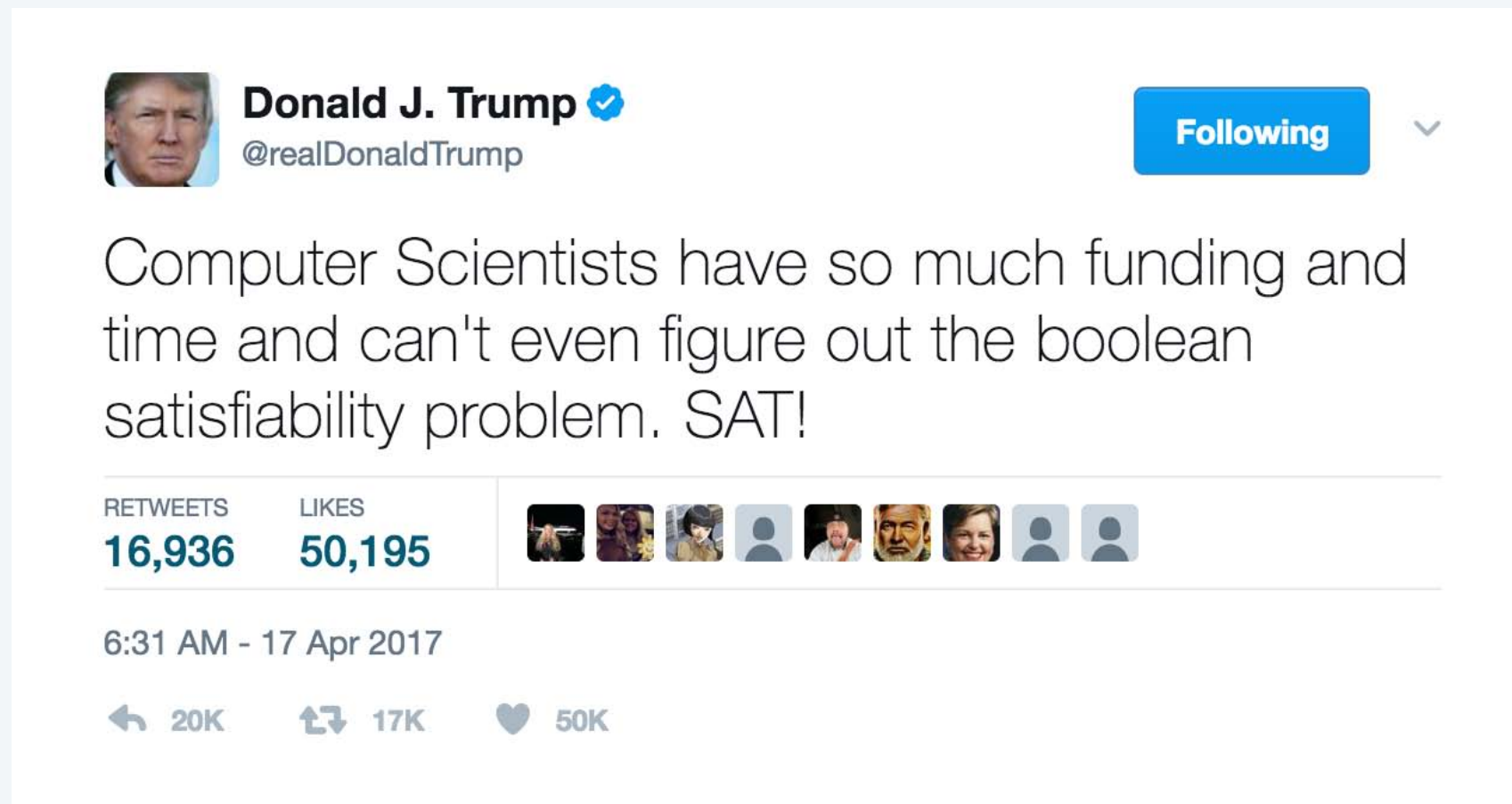
**Key application.** Electronic design automation (EDA).

# Satisfiability is hard

---

**Scientific hypothesis.** There does not exist a poly-time algorithm for 3-SAT.

**P vs. NP.** This hypothesis is equivalent to  $\mathbf{P} \neq \mathbf{NP}$  conjecture.



<https://www.facebook.com/pg/npcompleteteens>

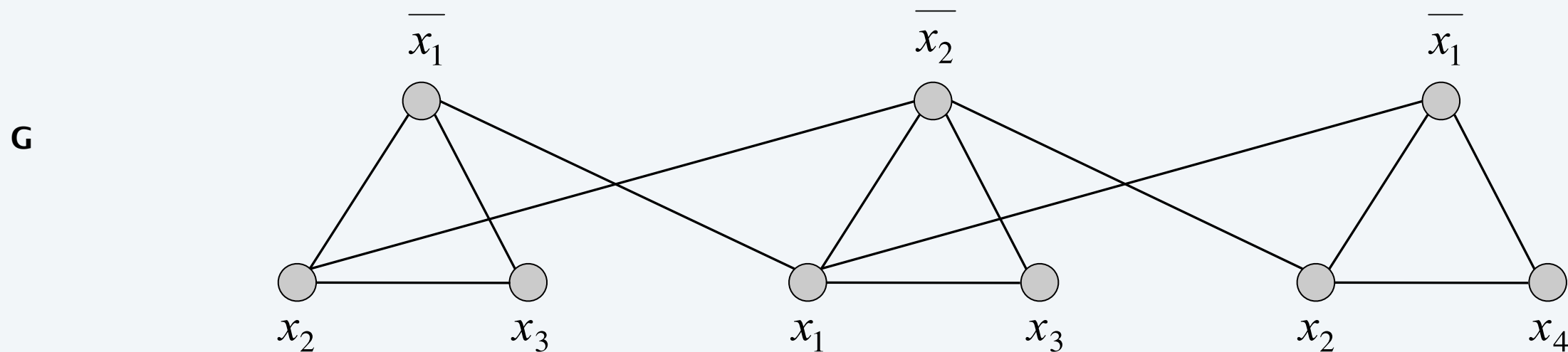
# 3-satisfiability reduces to independent set

**Theorem.**  $3\text{-SAT} \leq_P \text{INDEPENDENT-SET}$ .

**Pf.** 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.



**k = 3**

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

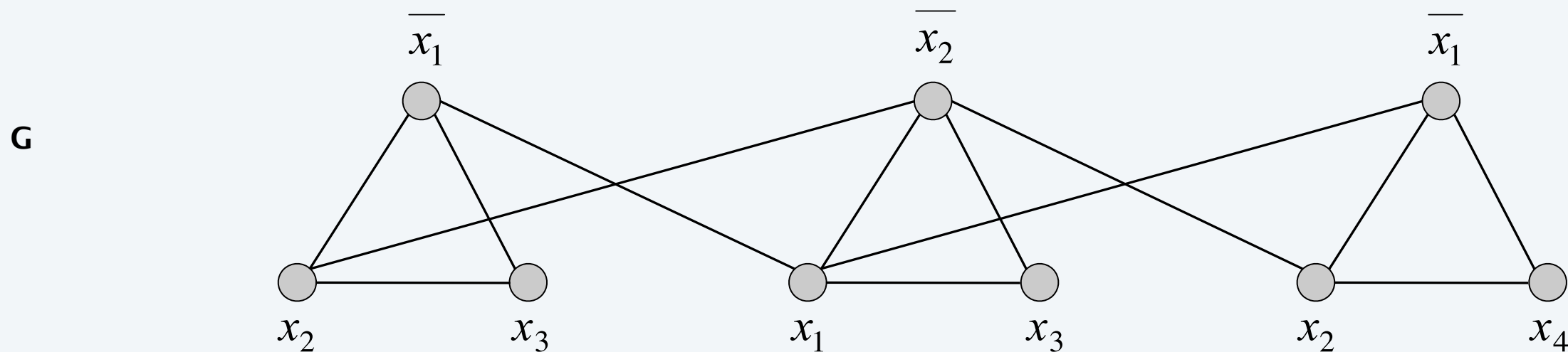
# 3-satisfiability reduces to independent set

**Lemma.**  $\Phi$  is satisfiable iff  $G$  contains an independent set of size  $k = |\Phi|$ .

**Pf.**  $\Rightarrow$  Consider any satisfying assignment for  $\Phi$ .

- Select one true literal from each clause/triangle.
- This is an independent set of size  $k = |\Phi|$ . ■

“yes” instances of 3-SAT  
are solved correctly



**k = 3**

$$\Phi = (\overline{x_1} \vee x_2 \vee x_3) \wedge (x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee x_2 \vee x_4)$$

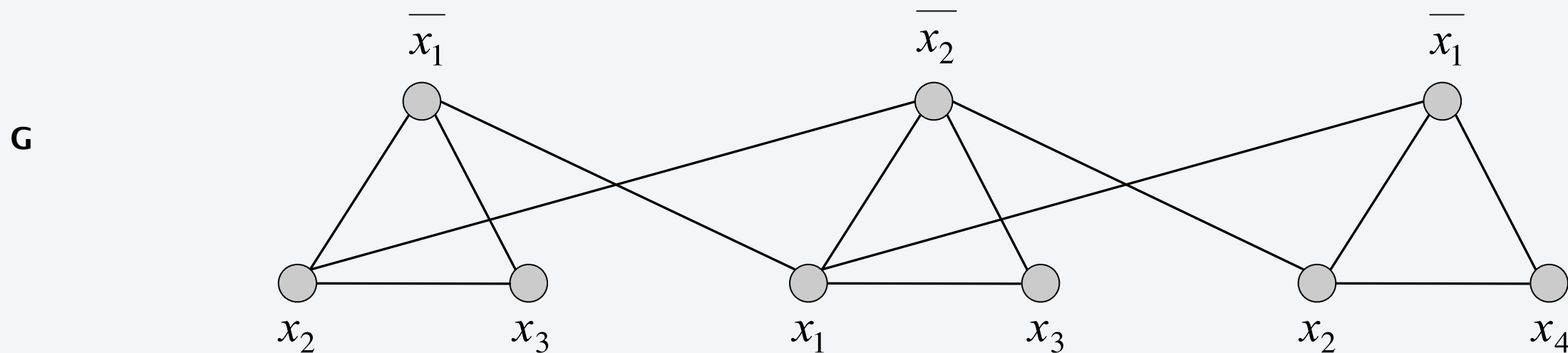
# 3-satisfiability reduces to independent set

**Lemma.**  $\Phi$  is satisfiable iff  $G$  contains an independent set of size  $k = |\Phi|$ .

**Pf.**  $\Leftarrow$  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. ■

“no” instances of 3-SAT  
are solved correctly



**k = 3**

$$\Phi = (\overline{x_1} \vee x_2 \vee x_3) \wedge (x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee x_2 \vee x_4)$$

# 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  $X \leq_p Z$ .

**Pf idea.** Compose the two algorithms.

**Ex.**  $3\text{-SAT} \leq_p \text{INDEPENDENT-SET} \leq_p \text{VERTEX-COVER} \leq_p \text{SET-COVER}$ .

# DECISION, SEARCH, AND OPTIMIZATION PROBLEMS

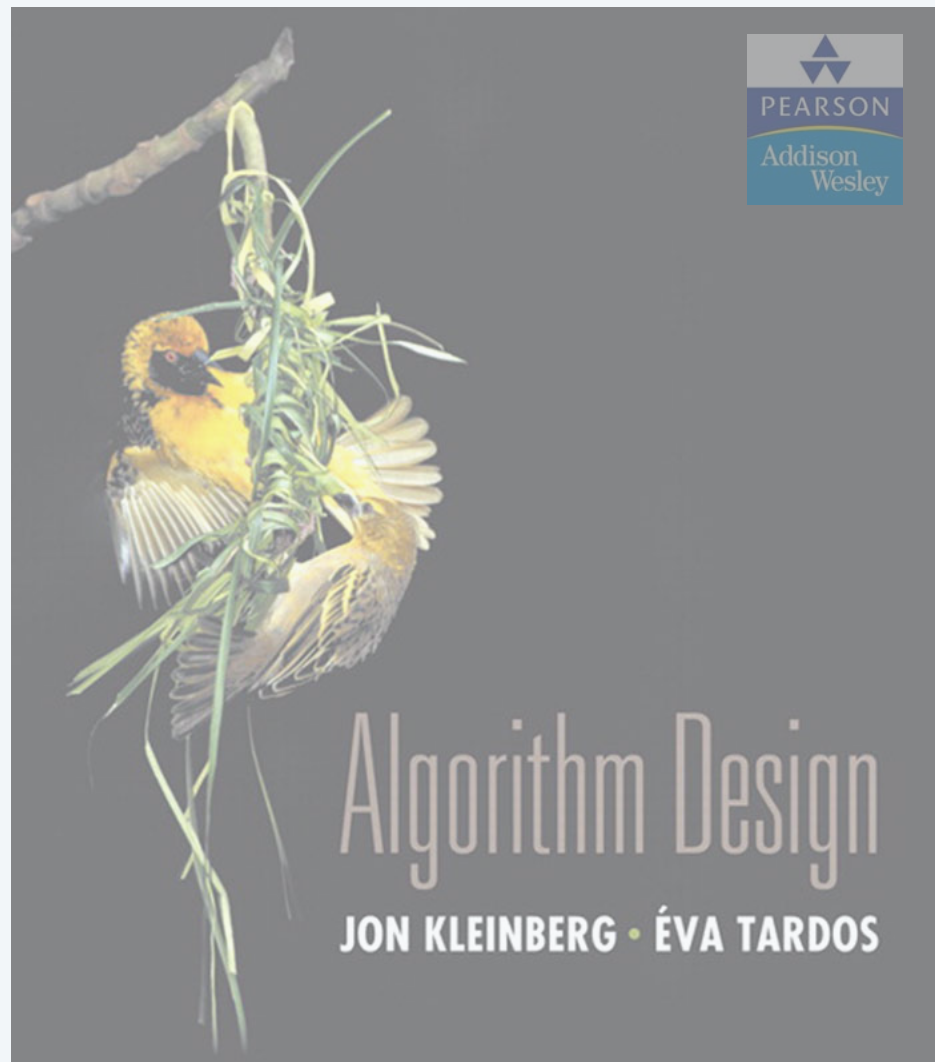


**Decision problem.** Does there **exist** a vertex cover of size  $\leq k$ ?

**Search problem.** **Find** a vertex cover of size  $\leq k$ .

**Optimization problem.** **Find** a vertex cover of **minimum** size.

**Goal.** Show that all three problems poly-time reduce to one another.



## SECTION 8.5

# 8. INTRACTABILITY I

---

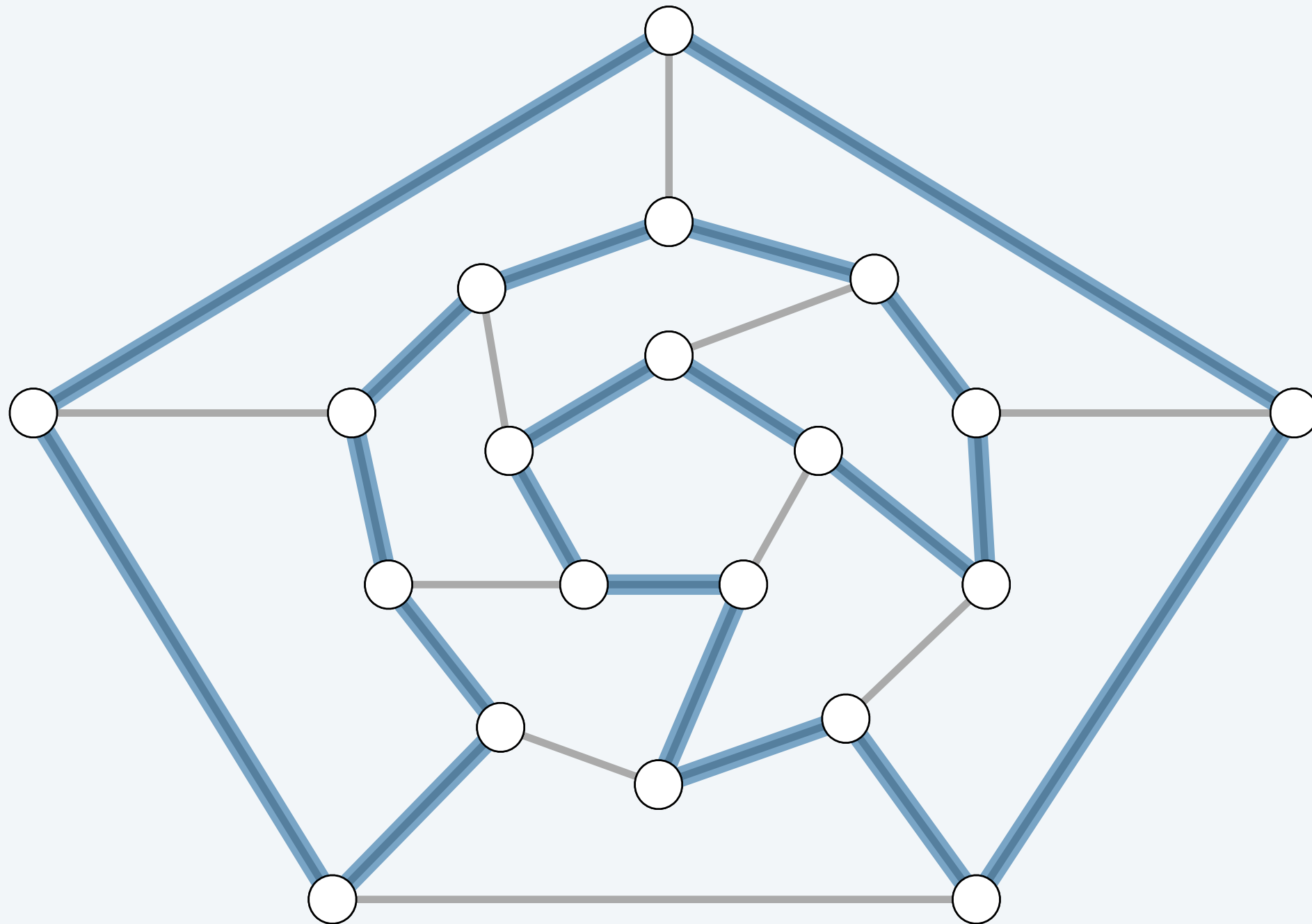
- ▶ *poly-time reductions*
- ▶ *packing and covering problems*
- ▶ *constraint satisfaction problems*
- ▶ ***sequencing problems***
- ▶ *partitioning problems*
- ▶ *graph coloring*
- ▶ *numerical problems*



# Hamilton cycle

---

**HAMILTON-CYCLE.** Given an undirected graph  $G = (V, E)$ , does there exist a cycle  $\Gamma$  that visits every node exactly once?

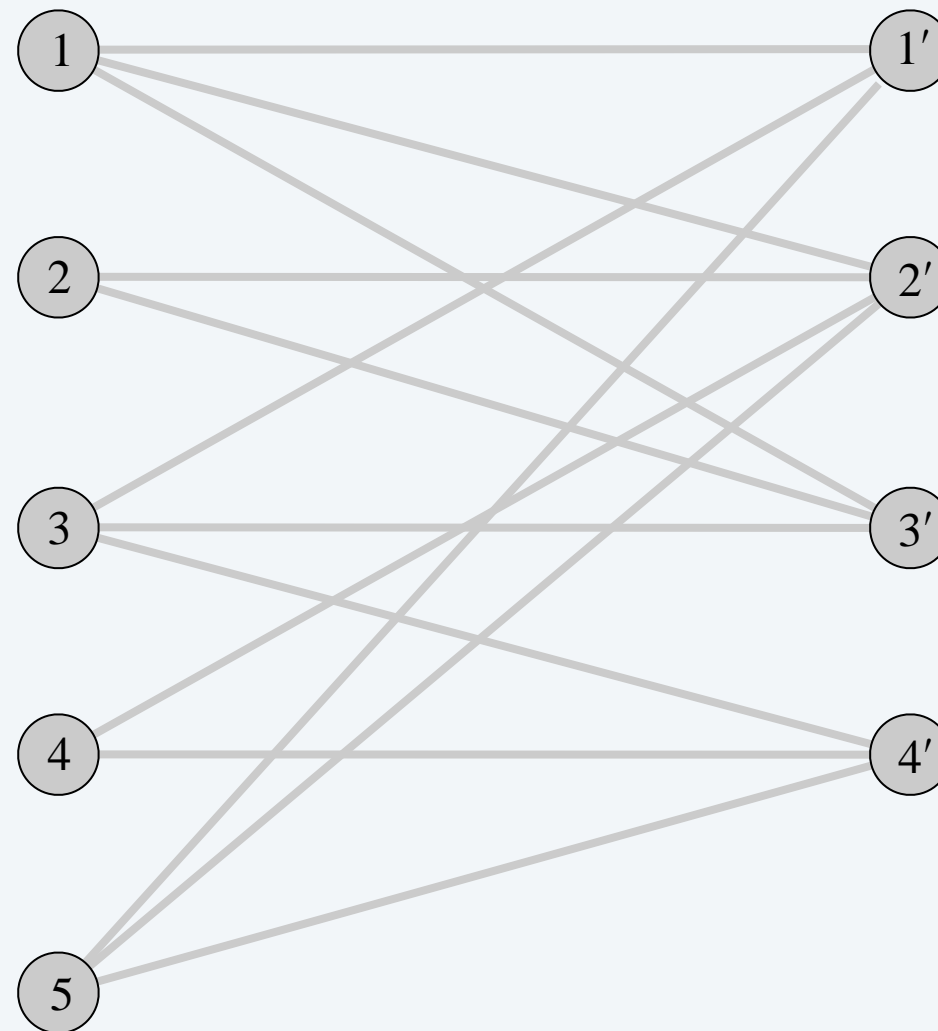


yes

# Hamilton cycle

---

**HAMILTON-CYCLE.** Given an undirected graph  $G = (V, E)$ , does there exist a cycle  $\Gamma$  that visits every node exactly once?



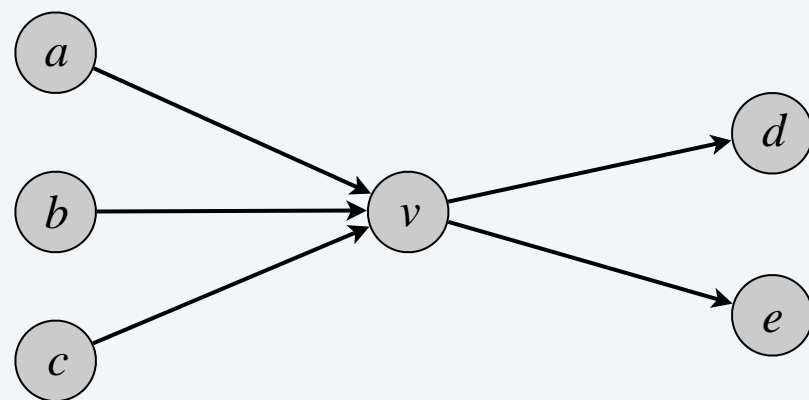
**no**

# Directed Hamilton cycle reduces to Hamilton cycle

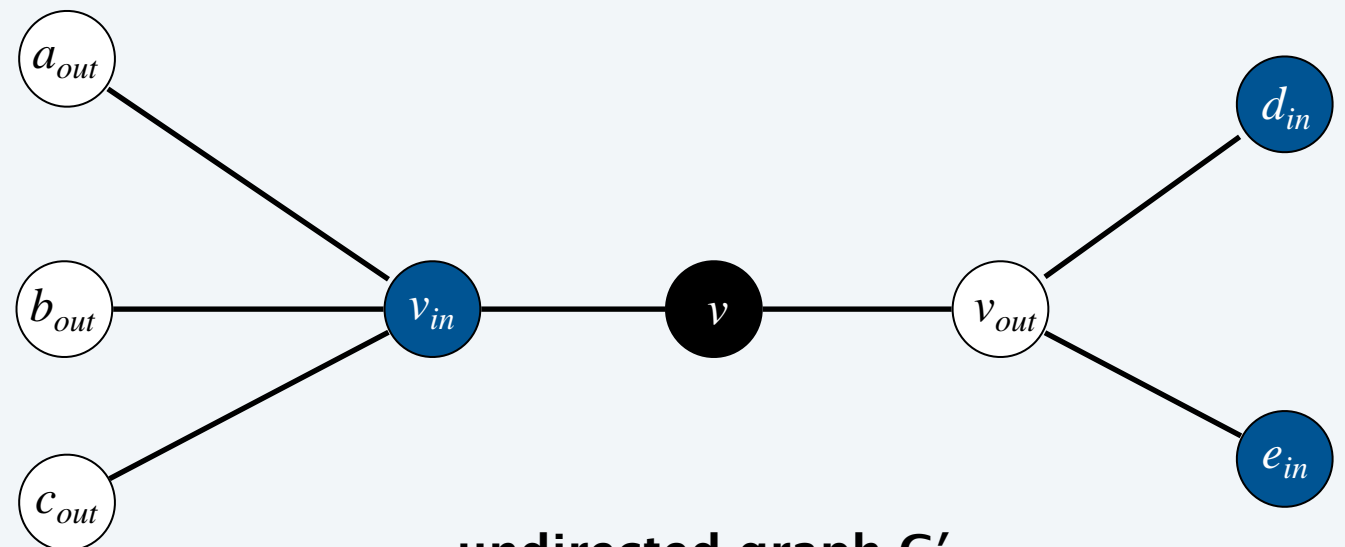
**DIRECTED-HAMILTON-CYCLE.** Given a directed graph  $G = (V, E)$ , does there exist a directed cycle  $\Gamma$  that visits every node exactly once?

**Theorem.**  $\text{DIRECTED-HAMILTON-CYCLE} \leq_p \text{HAMILTON-CYCLE}$ .

**Pf.** Given a directed graph  $G = (V, E)$ , construct a graph  $G'$  with  $3n$  nodes.



directed graph  $G$



undirected graph  $G'$

# Directed Hamilton cycle reduces to Hamilton cycle

---

**Lemma.**  $G$  has a directed Hamilton cycle iff  $G'$  has a Hamilton cycle.

**Pf.**  $\Rightarrow$

- Suppose  $G$  has a directed Hamilton cycle  $\Gamma$ .
- Then  $G'$  has an undirected Hamilton cycle (same order). ■

**Pf.**  $\Leftarrow$

- Suppose  $G'$  has an undirected Hamilton cycle  $\Gamma'$ .
- $\Gamma'$  must visit nodes in  $G'$  using one of following two orders:  
 $\dots, \textit{black}, \textit{white}, \textit{blue}, \textit{black}, \textit{white}, \textit{blue}, \textit{black}, \textit{white}, \textit{blue}, \dots$   
 $\dots, \textit{black}, \textit{blue}, \textit{white}, \textit{black}, \textit{blue}, \textit{white}, \textit{black}, \textit{blue}, \textit{white}, \dots$
- Black nodes in  $\Gamma'$  comprise either a directed Hamilton cycle  $\Gamma$  in  $G$ , or reverse of one. ■

## 3-satisfiability reduces to directed Hamilton cycle

---

**Theorem.**  $3\text{-SAT} \leq_P \text{DIRECTED-HAMILTON-CYCLE}$ .

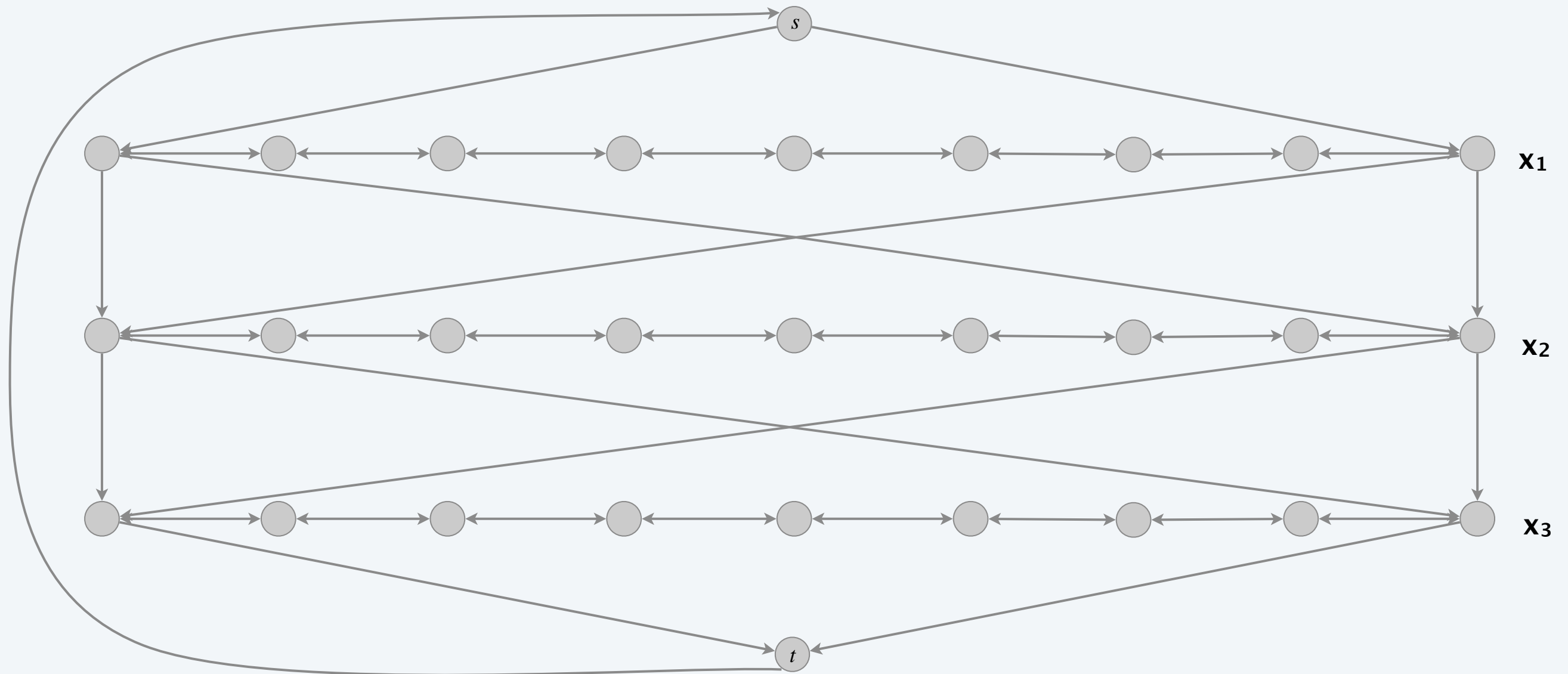
**Pf.** Given an instance  $\Phi$  of 3-SAT, we construct an instance  $G$  of DIRECTED-HAMILTON-CYCLE that has a Hamilton cycle iff  $\Phi$  is satisfiable.

**Construction overview.** Let  $n$  denote the number of variables in  $\Phi$ . We will construct a graph  $G$  that has  $2^n$  Hamilton cycles, with each cycle corresponding to one of the  $2^n$  possible truth assignments.

# 3-satisfiability reduces to directed Hamilton cycle

**Construction.** Given 3-SAT instance  $\Phi$  with  $n$  variables  $x_i$  and  $k$  clauses.

- Construct  $G$  to have  $2^n$  Hamilton cycles.
- Intuition: traverse path  $i$  from left to right  $\Leftrightarrow$  set variable  $x_i = \text{true}$ .





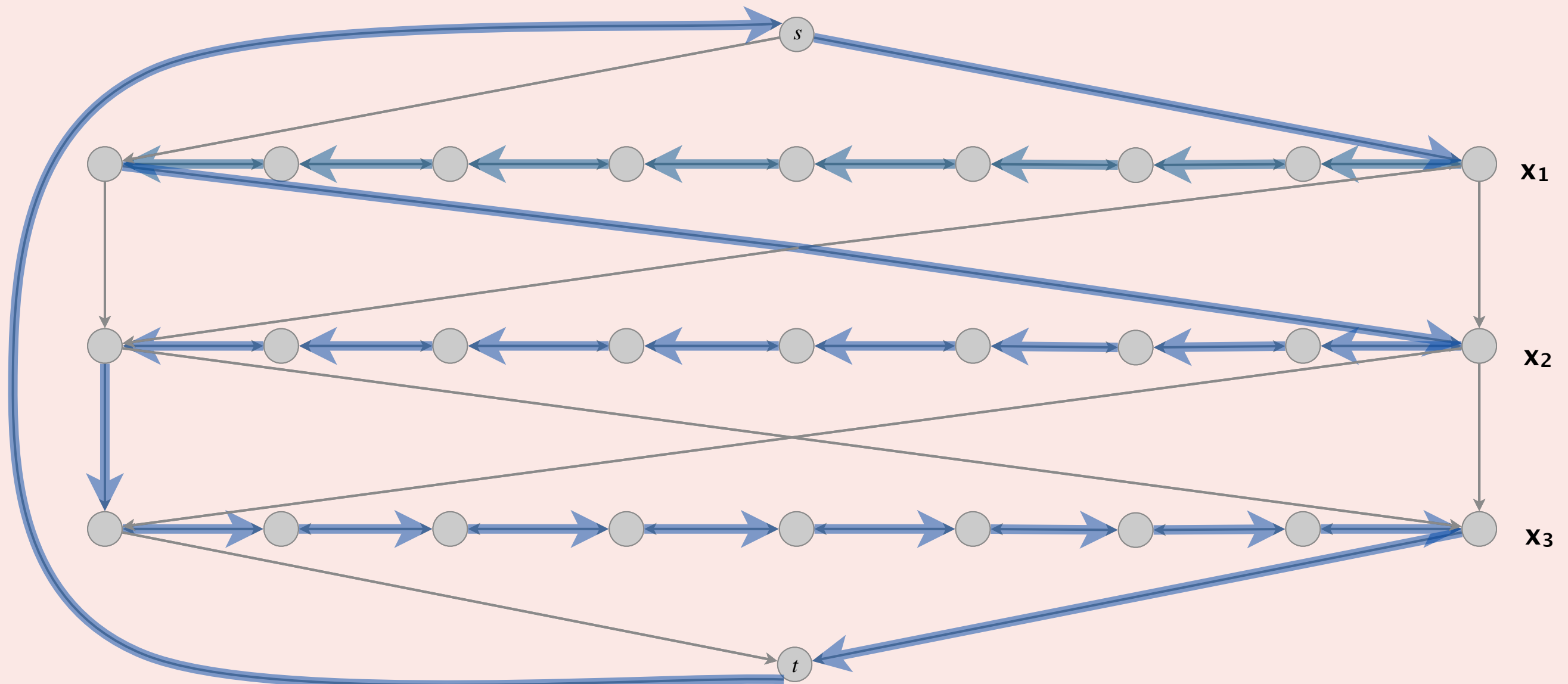
Which is truth assignment corresponding to Hamilton cycle below?

**A.**  $x_1 = \text{true}, x_2 = \text{true}, x_3 = \text{true}$

**C.**  $x_1 = \text{false}, x_2 = \text{false}, x_3 = \text{true}$

**B.**  $x_1 = \text{true}, x_2 = \text{true}, x_3 = \text{false}$

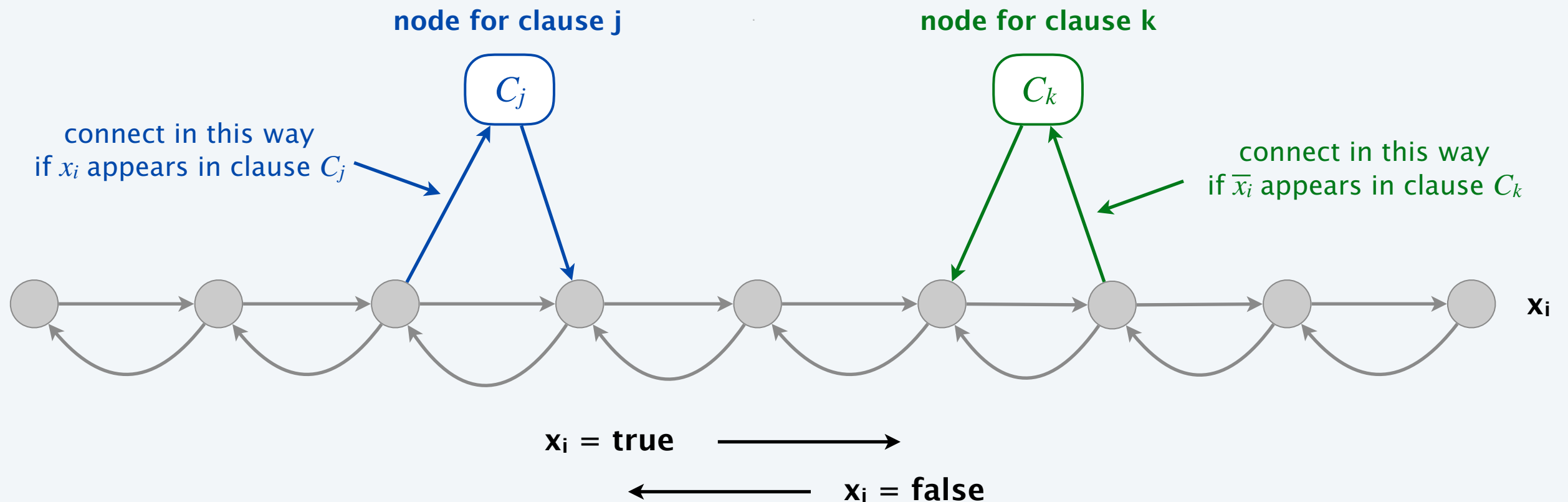
**D.**  $x_1 = \text{false}, x_2 = \text{false}, x_3 = \text{false}$



# 3-satisfiability reduces to directed Hamilton cycle

**Construction.** Given 3-SAT instance  $\Phi$  with  $n$  variables  $x_i$  and  $k$  clauses.

- For each clause: add a node and 2 edges per literal.

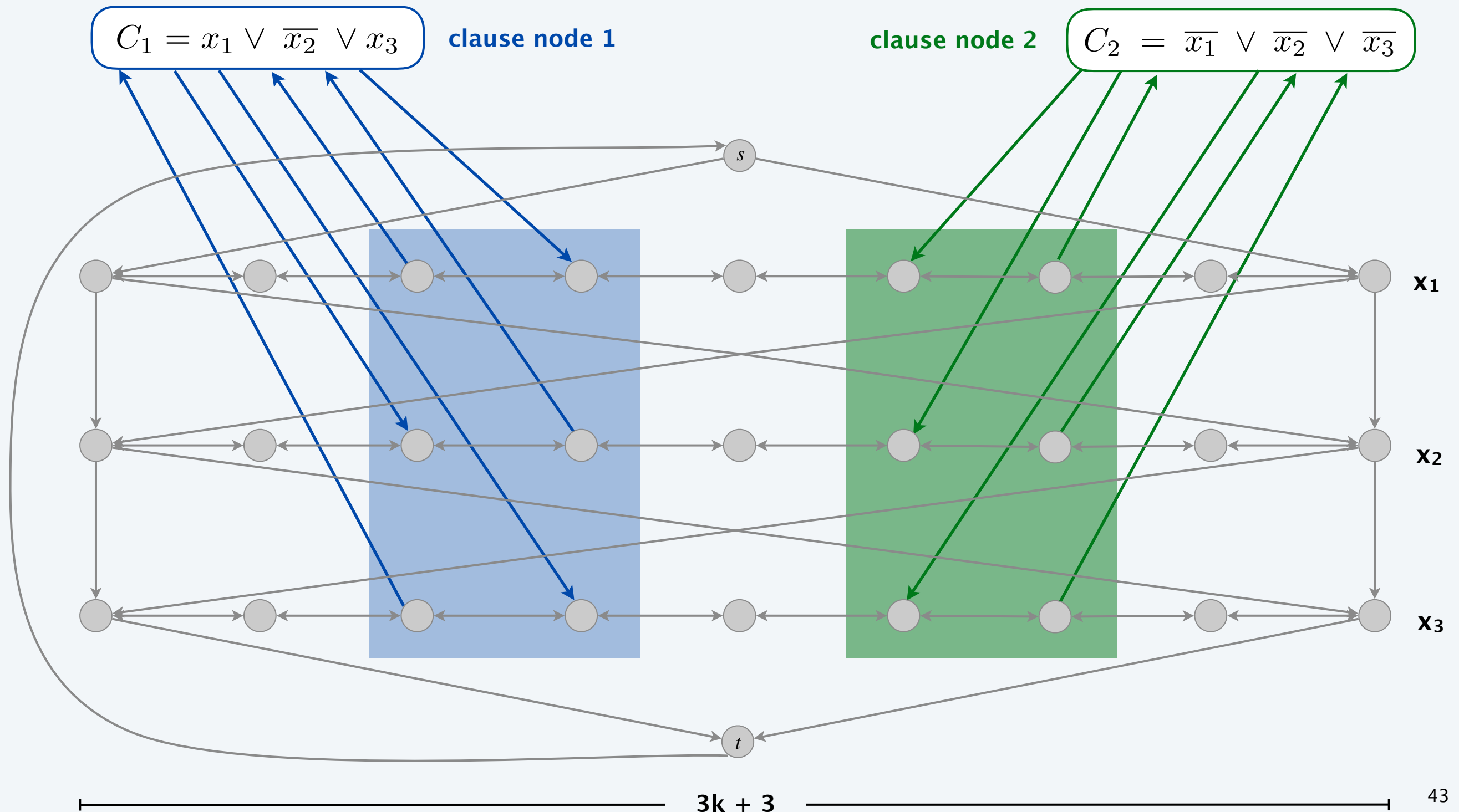




# 3-satisfiability reduces to directed Hamilton cycle

**Construction.** Given 3-SAT instance  $\Phi$  with  $n$  variables  $x_i$  and  $k$  clauses.

- For each clause: add a node and 2 edges per literal.



## 3-satisfiability reduces to directed Hamilton cycle

---

**Lemma.**  $\Phi$  is satisfiable iff  $G$  has a Hamilton cycle.

**Pf.**  $\Rightarrow$

- Suppose 3-SAT instance  $\Phi$  has satisfying assignment  $x^*$ .
- Then, define Hamilton cycle  $\Gamma$  in  $G$  as follows:
  - if  $x_i^* = \text{true}$ , traverse row  $i$  from left to right
  - if  $x_i^* = \text{false}$ , traverse row  $i$  from right to left
  - for each clause  $C_j$ , there will be at least one row  $i$  in which we are going in “correct” direction to splice clause node  $C_j$  into cycle (and we splice in  $C_j$  exactly once) ■

## 3-satisfiability reduces to directed Hamilton cycle

---

**Lemma.**  $\Phi$  is satisfiable iff  $G$  has a Hamilton cycle.

**Pf.**  $\Leftarrow$

- Suppose  $G$  has a Hamilton cycle  $\Gamma$ .
- If  $\Gamma$  enters clause node  $C_j$ , it must depart on mate edge.
  - nodes immediately before and after  $C_j$  are connected by an edge  $e \in E$
  - removing  $C_j$  from cycle, and replacing it with edge  $e$  yields Hamilton cycle on  $G - \{ C_j \}$
- Continuing in this way, we are left with a Hamilton cycle  $\Gamma'$  in  $G - \{ C_1, C_2, \dots, C_k \}$ .
- Set  $x_i^* = \text{true}$  if  $\Gamma'$  traverses row  $i$  left-to-right; otherwise, set  $x_i^* = \text{false}$ .
- traversed in “correct” direction, and each clause is satisfied. ■

# Poly-time reductions

---

constraint satisfaction

3-SAT

3-SAT poly-time reduces  
to INDEPENDENT-SET

INDEPENDENT-SET

DIR-HAM-CYCLE

3-COLOR

SUBSET-SUM

VERTEX-COVER

HAM-CYCLE

KNAPSACK

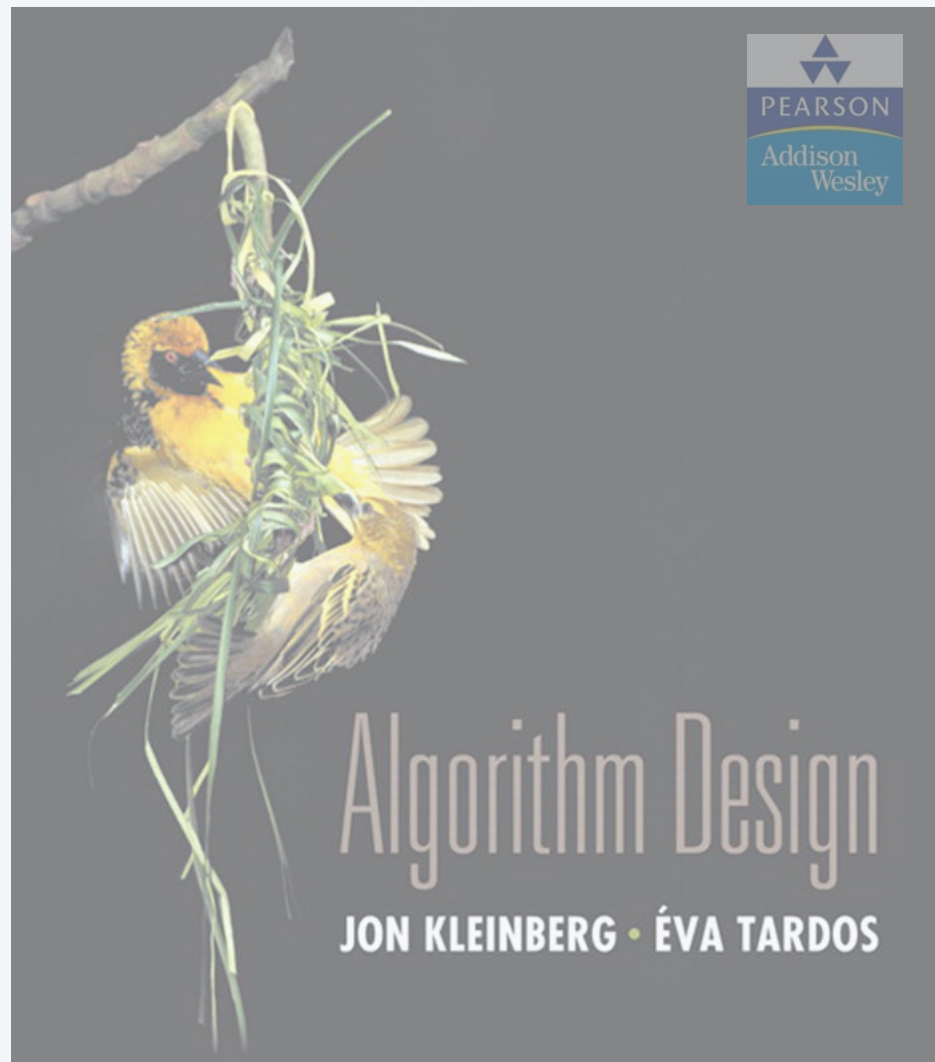
SET-COVER

packing and covering

sequencing

partitioning

numerical

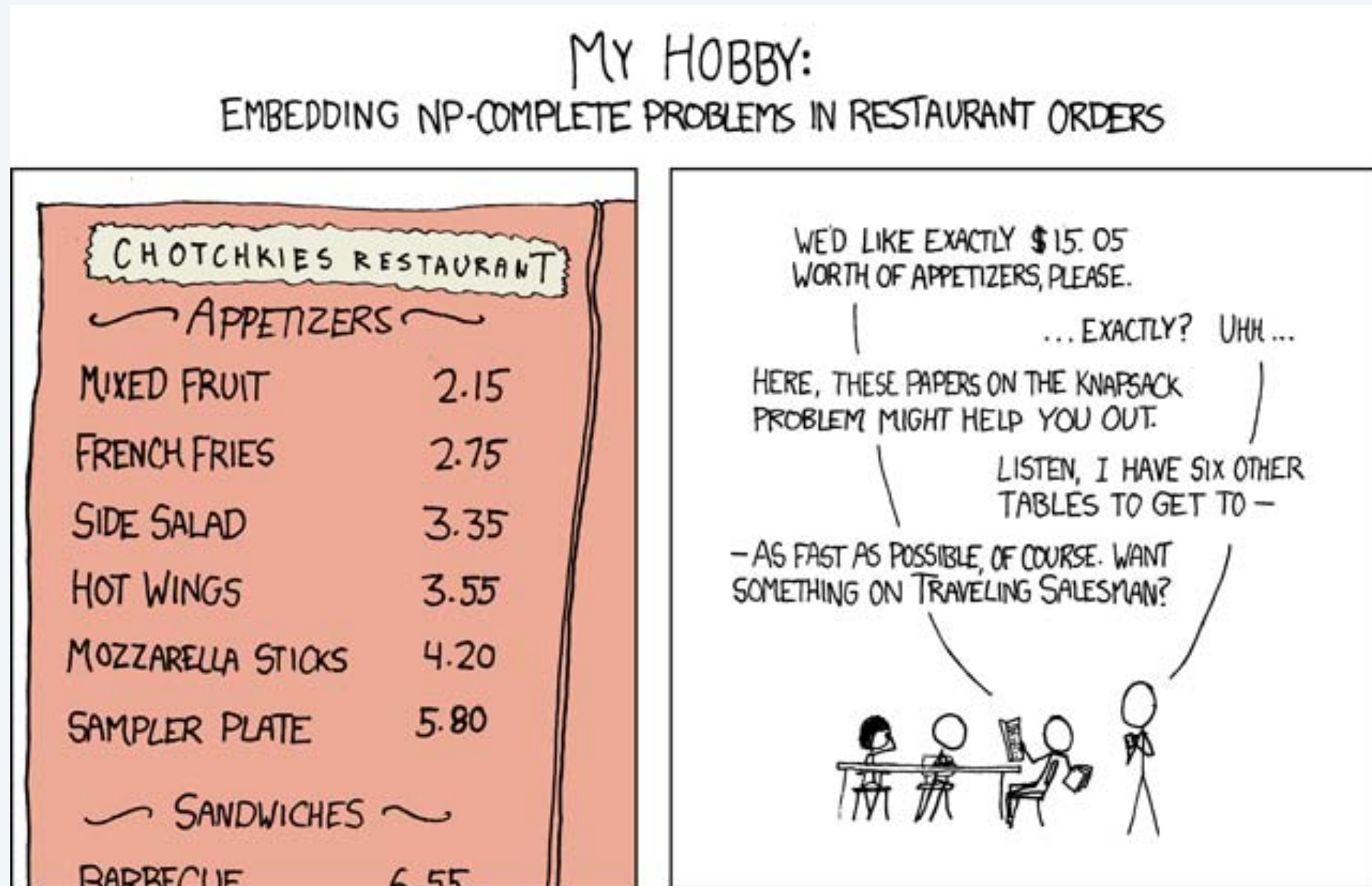


## SECTION 8.8

# 8. INTRACTABILITY I

---

- ▶ *poly-time reductions*
- ▶ *packing and covering problems*
- ▶ *constraint satisfaction problems*
- ▶ *sequencing problems*
- ▶ *partitioning problems*
- ▶ *graph coloring*
- ▶ ***numerical problems***



NP-Complete by Randall Munro

<http://xkcd.com/287>

Creative Commons Attribution-NonCommercial 2.5

# Subset sum

---

**SUBSET-SUM.** Given  $n$  natural numbers  $w_1, \dots, w_n$  and an integer  $W$ , is there a subset that adds up to exactly  $W$ ?

**Ex.** { 215, 215, 275, 275, 355, 355, 420, 420, 580, 580, 655, 655 },  $W = 1505$ .

**Yes.**  $215 + 355 + 355 + 580 = 1505$ .

**Remark.** With arithmetic problems, input integers are encoded in binary. Poly-time reduction must be polynomial in **binary** encoding.

# Subset sum

---

**Theorem.**  $3\text{-SAT} \leq_p \text{SUBSET-SUM}$ .

**Pf.** Given an instance  $\Phi$  of 3-SAT, we construct an instance of SUBSET-SUM that has solution iff  $\Phi$  is satisfiable.



# 3-satisfiability reduces to subset sum

**Construction.** Given 3-SAT instance  $\Phi$  with  $n$  variables and  $k$  clauses, form  $2n + 2k$  decimal integers, each having  $n + k$  digits:

- Include one digit for each variable  $x_i$  and one digit for each clause  $C_j$ .
- Include two numbers for each variable  $x_i$ .
- Include two numbers for each clause  $C_j$ .
- Sum of each  $x_i$  digit is 1;  
sum of each  $C_j$  digit is 4.

**Key property.** No carries possible  $\Rightarrow$  each digit yields one equation.

$C_1 =$	$\neg x_1$	$\vee$	$x_2$	$\vee$	$x_3$
$C_2 =$	$x_1$	$\vee$	$\neg x_2$	$\vee$	$x_3$
$C_3 =$	$\neg x_1$	$\vee$	$\neg x_2$	$\vee$	$\neg x_3$

3-SAT instance

dummies to get clause columns to sum to 4

	$x_1$	$x_2$	$x_3$	$C_1$	$C_2$	$C_3$	
$x_1$	1	0	0	0	1	0	100,010
$\neg x_1$	1	0	0	1	0	1	100,101
$x_2$	0	1	0	1	0	0	10,100
$\neg x_2$	0	1	0	0	1	1	10,011
$x_3$	0	0	1	1	1	0	1,110
$\neg x_3$	0	0	1	0	0	1	1,001
}	0	0	0	1	0	0	100
	0	0	0	2	0	0	200
	0	0	0	0	1	0	10
	0	0	0	0	2	0	20
	0	0	0	0	0	1	1
	0	0	0	0	0	2	2
$W$	1	1	1	4	4	4	111,444

SUBSET-SUM instance

# 3-satisfiability reduces to subset sum

**Lemma.**  $\Phi$  is satisfiable iff there exists a subset that sums to  $W$ .

**Pf.**  $\Rightarrow$  Suppose 3-SAT instance  $\Phi$  has satisfying assignment  $x^*$ .

- If  $x_i^* = true$ , select integer in row  $x_i$ ;  
otherwise, select integer in row  $\neg x_i$ .
- Each  $x_i$  digit sums to 1.
- Since  $\Phi$  is satisfiable, each  $C_j$  digit sums to at least 1 from  $x_i$  and  $\neg x_i$  rows.
- Select dummy integers to make  $C_j$  digits sum to 4. ■

$C_1 =$	$\neg x_1$	$\vee$	$x_2$	$\vee$	$x_3$
$C_2 =$	$x_1$	$\vee$	$\neg x_2$	$\vee$	$x_3$
$C_3 =$	$\neg x_1$	$\vee$	$\neg x_2$	$\vee$	$\neg x_3$

3-SAT instance

dummies to get clause columns to sum to 4

	$x_1$	$x_2$	$x_3$	$C_1$	$C_2$	$C_3$	
$x_1$	1	0	0	0	1	0	100,010
$\neg x_1$	1	0	0	1	0	1	100,101
$x_2$	0	1	0	1	0	0	10,100
$\neg x_2$	0	1	0	0	1	1	10,011
$x_3$	0	0	1	1	1	0	1,110
$\neg x_3$	0	0	1	0	0	1	1,001
<div> </div>	0	0	0	1	0	0	100
	0	0	0	2	0	0	200
	0	0	0	0	1	0	10
	0	0	0	0	2	0	20
	0	0	0	0	0	1	1
	0	0	0	0	0	2	2
$W$	1	1	1	4	4	4	111,444

SUBSET-SUM instance

# 3-satisfiability reduces to subset sum

**Lemma.**  $\Phi$  is satisfiable iff there exists a subset that sums to  $W$ .

**Pf.**  $\Leftarrow$  Suppose there exists a subset  $S^*$  that sums to  $W$ .

- Digit  $x_i$  forces subset  $S^*$  to select either row  $x_i$  or row  $\neg x_i$  (but not both).
- If row  $x_i$  selected, assign  $x_i^* = true$  ; otherwise, assign  $x_i^* = false$ .

Digit  $C_j$  forces subset  $S^*$  to select at least one literal in clause. ■

$C_1$	=	$\neg x_1$	$\vee$	$x_2$	$\vee$	$x_3$
$C_2$	=	$x_1$	$\vee$	$\neg x_2$	$\vee$	$x_3$
$C_3$	=	$\neg x_1$	$\vee$	$\neg x_2$	$\vee$	$\neg x_3$

3-SAT instance

dummies to get clause columns to sum to 4

	$x_1$	$x_2$	$x_3$	$C_1$	$C_2$	$C_3$	
$x_1$	1	0	0	0	1	0	100,010
$\neg x_1$	1	0	0	1	0	1	100,101
$x_2$	0	1	0	1	0	0	10,100
$\neg x_2$	0	1	0	0	1	1	10,011
$x_3$	0	0	1	1	1	0	1,110
$\neg x_3$	0	0	1	0	0	1	1,001
}	0	0	0	1	0	0	100
	0	0	0	2	0	0	200
	0	0	0	0	1	0	10
	0	0	0	0	2	0	20
	0	0	0	0	0	1	1
	0	0	0	0	0	2	2
$W$	1	1	1	4	4	4	111,444

SUBSET-SUM instance

# SUBSET SUM REDUCES TO KNAPSACK



**SUBSET-SUM.** Given  $n$  natural numbers  $w_1, \dots, w_n$  and an integer  $W$ , is there a subset that adds up to exactly  $W$ ?

**KNAPSACK.** Given a set of items  $X$ , weights  $u_i \geq 0$ , values  $v_i \geq 0$ , a weight limit  $U$ , and a target value  $V$ , is there a subset  $S \subseteq X$  such that:

$$\sum_{i \in S} u_i \leq U, \quad \sum_{i \in S} v_i \geq V$$

**Recall.**  $O(n U)$  dynamic programming algorithm for KNAPSACK.

**Challenge.** Prove  $\text{SUBSET-SUM} \leq_P \text{KNAPSACK}$ .

**Pf.** Given instance  $(w_1, \dots, w_n, W)$  of SUBSET-SUM, create KNAPSACK instance:

# Poly-time reductions

---

constraint satisfaction

3-SAT

3-SAT poly-time reduces  
to INDEPENDENT-SET

INDEPENDENT-SET

DIR-HAM-CYCLE

3-COLOR

SUBSET-SUM

VERTEX-COVER

HAM-CYCLE

KNAPSACK

SET-COVER

packing and covering

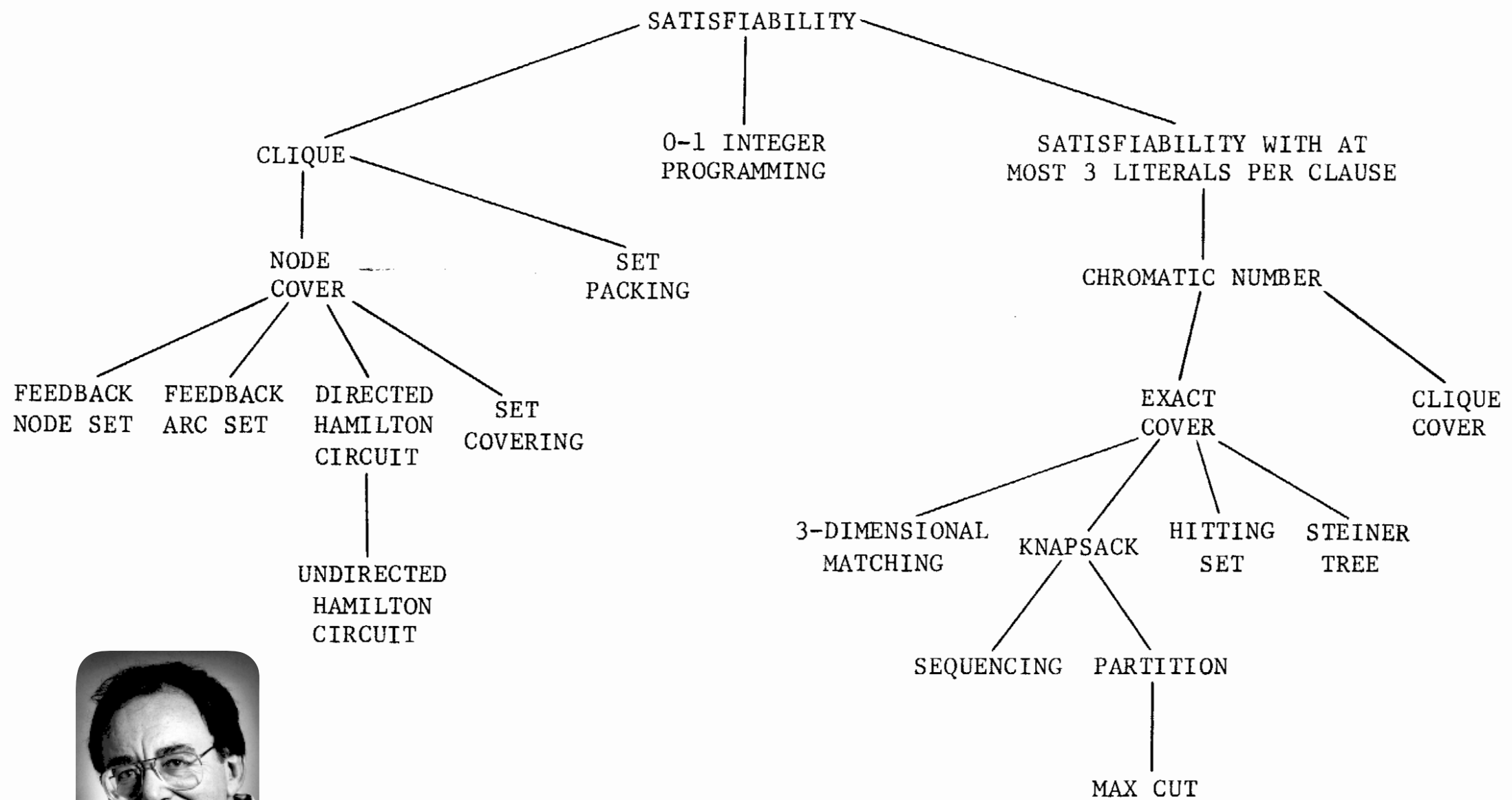
sequencing

partitioning

numerical

# Karp's 20 poly-time reductions from satisfiability

96



Dick Karp (1972)  
1985 Turing Award

FIGURE 1 Complete Problems

RICHARD M. KARP