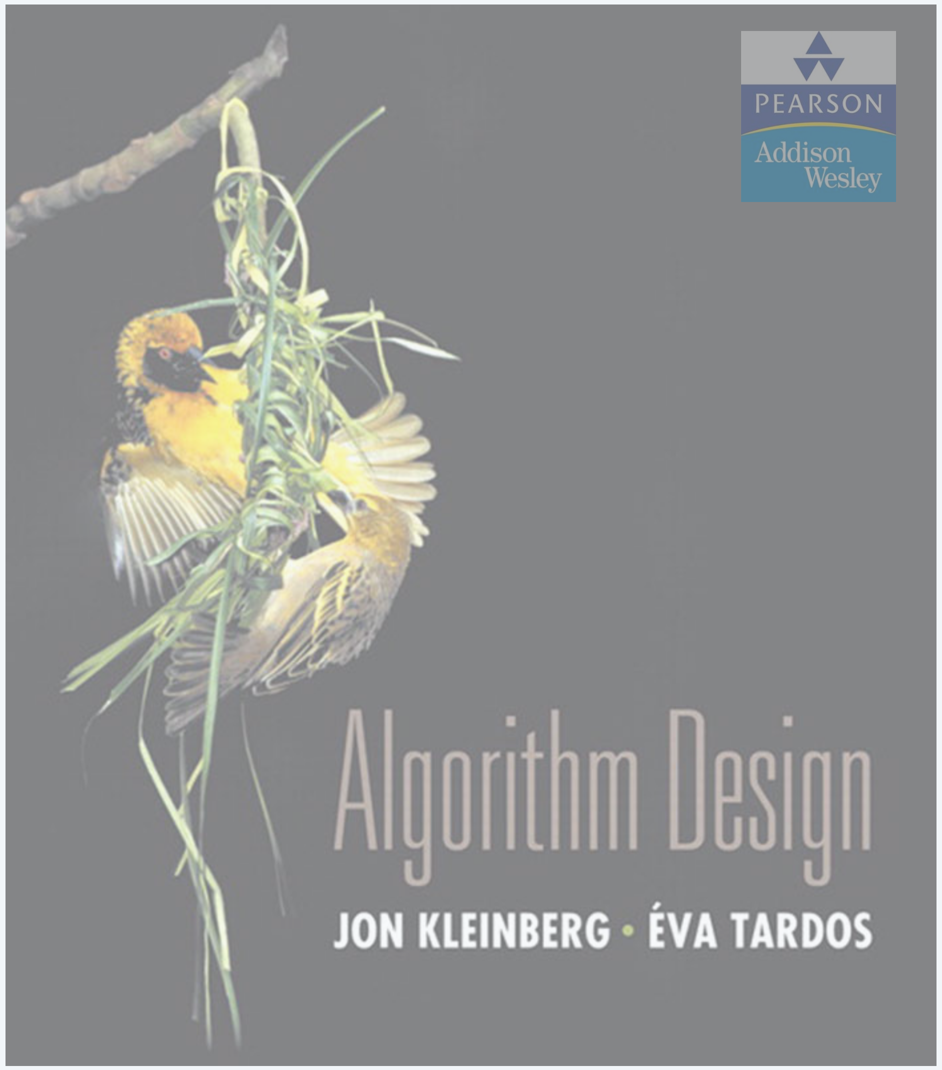


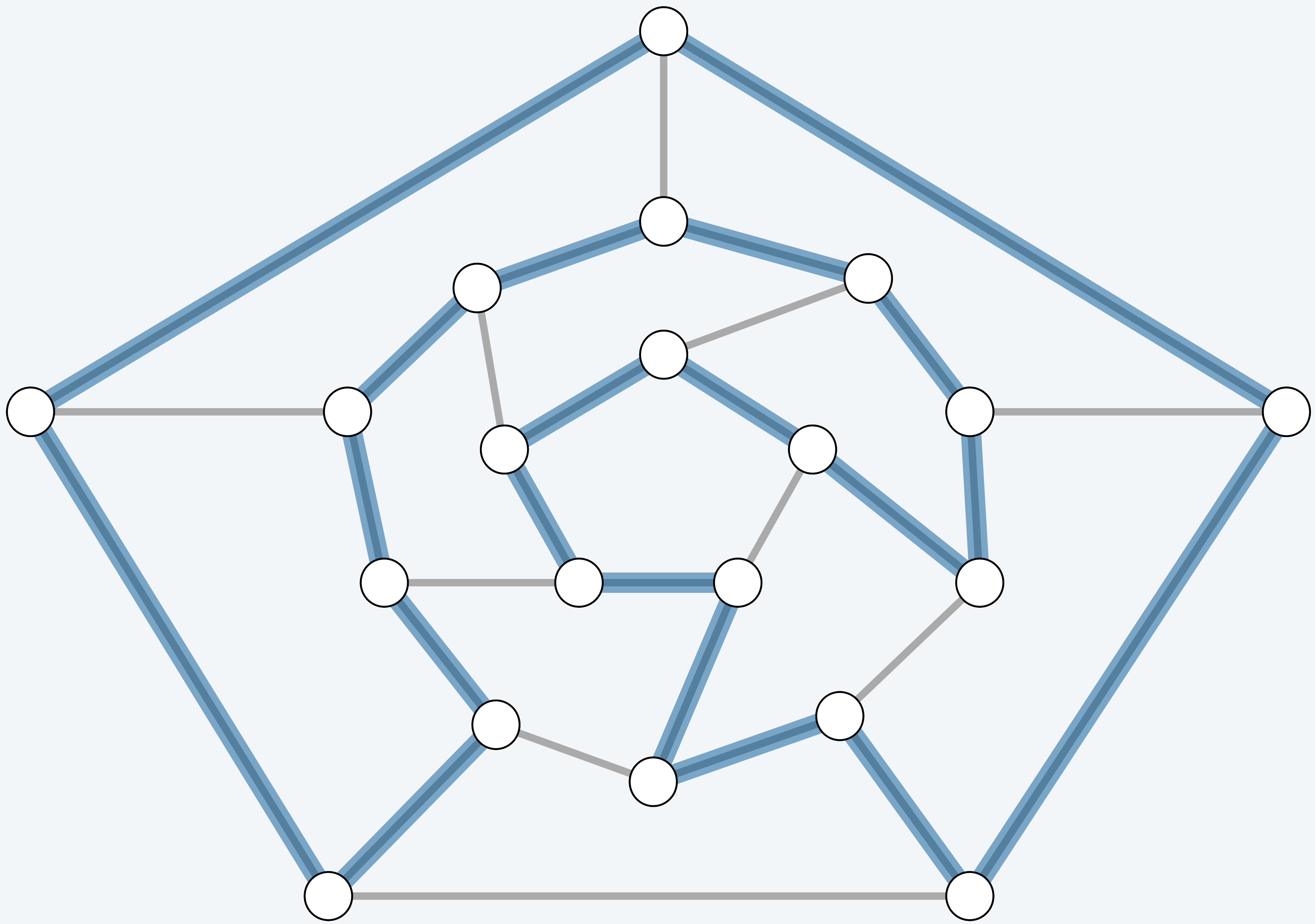
SECTION 8.5

8. INTRACTABILITY I

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

Hamilton cycle

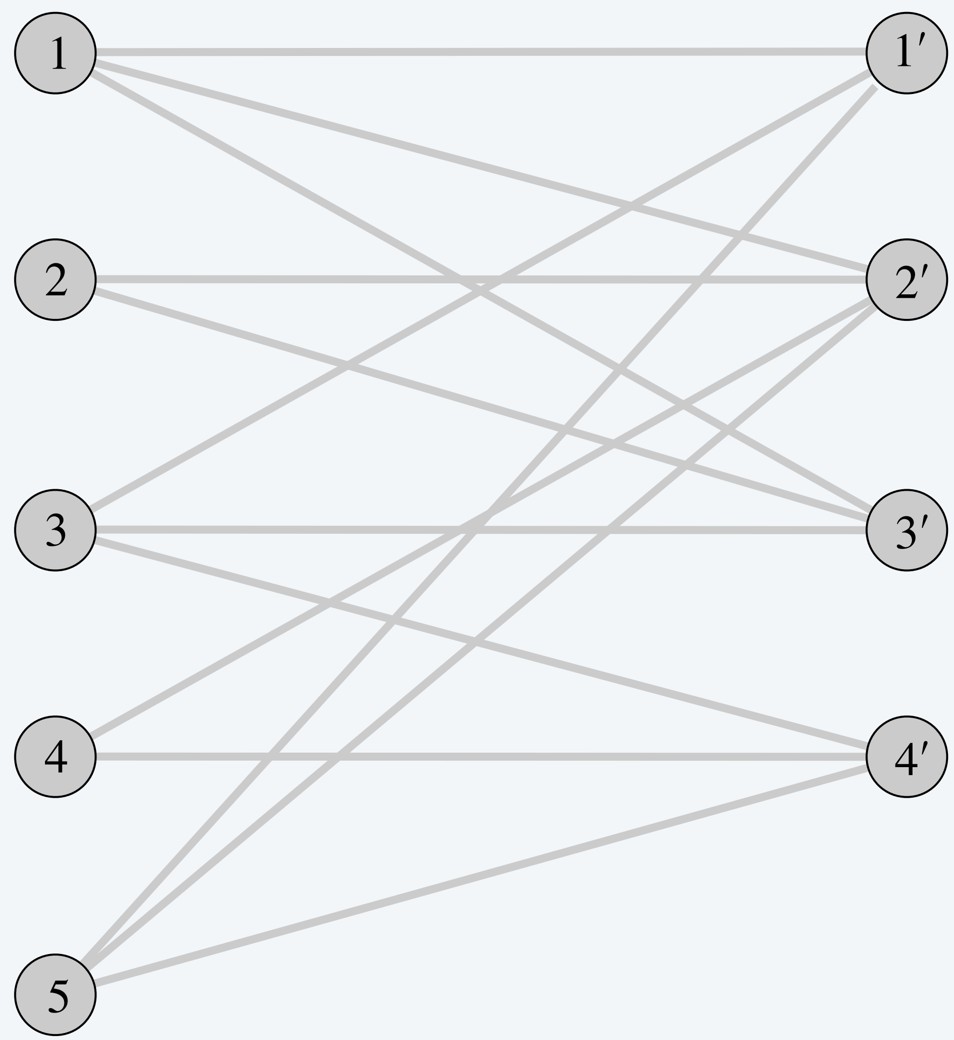
HAMILTON-CYCLE. Given an undirected graph $G = (V, E)$, does there exist a cycle Γ that visits every node exactly once?



yes

Hamilton cycle

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



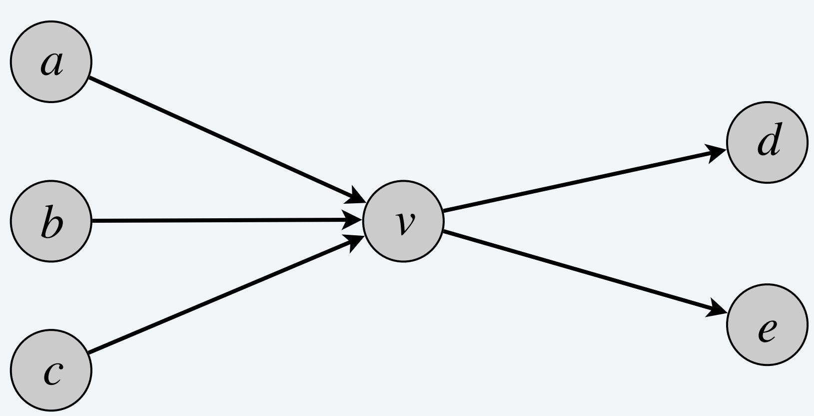
no

Directed Hamilton cycle reduces to Hamilton cycle

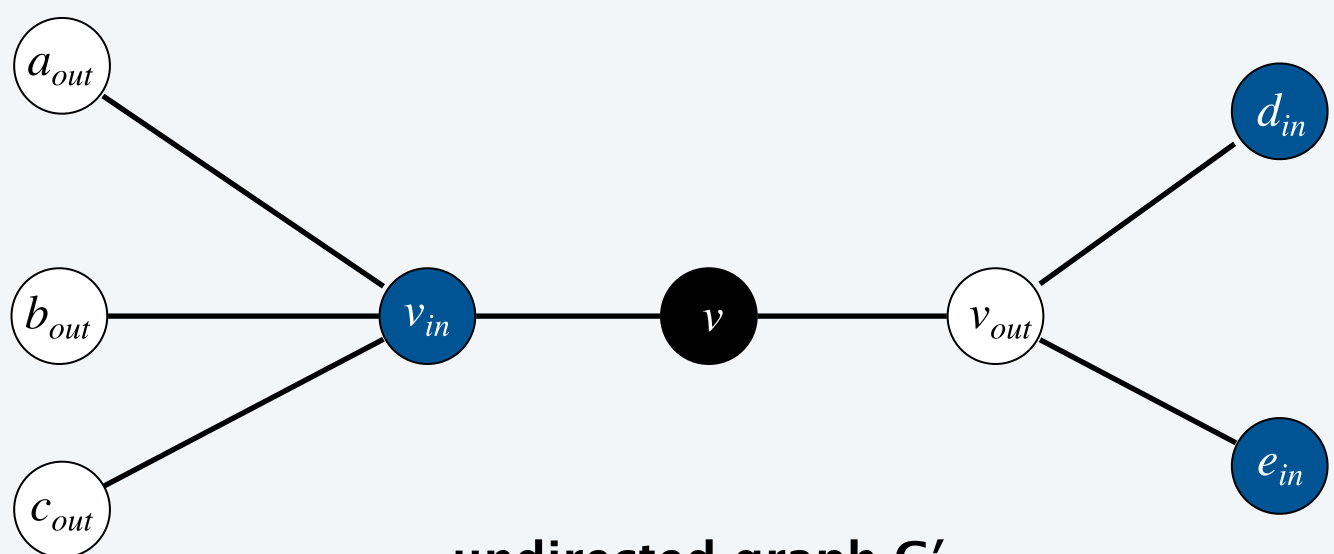
DIRECTED-HAMILTON-CYCLE. Given a directed graph $G = (V, E)$, does there exist a directed cycle Γ that visits every node exactly once?

Theorem. DIRECTED-HAMILTON-CYCLE $\le_P$ 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 Γ.
- Then G' has an undirected Hamilton cycle (same order). ▪

Pf. $\Leftarrow$

- Suppose G' has an undirected Hamilton cycle Γ'.
- Γ' must visit nodes in G' using one of following two orders:

  $\ldots, black, white, blue, black, white, blue, black, white, blue, \ldots$

  $\ldots, black, blue, white, black, blue, white, black, blue, white, \ldots$
- Black nodes in Γ' comprise either a directed Hamilton cycle Γ in G, or reverse of one. ▪

3-satisfiability reduces to directed Hamilton cycle

Theorem. 3-SAT $\leq_P$ DIRECTED-HAMILTON-CYCLE.

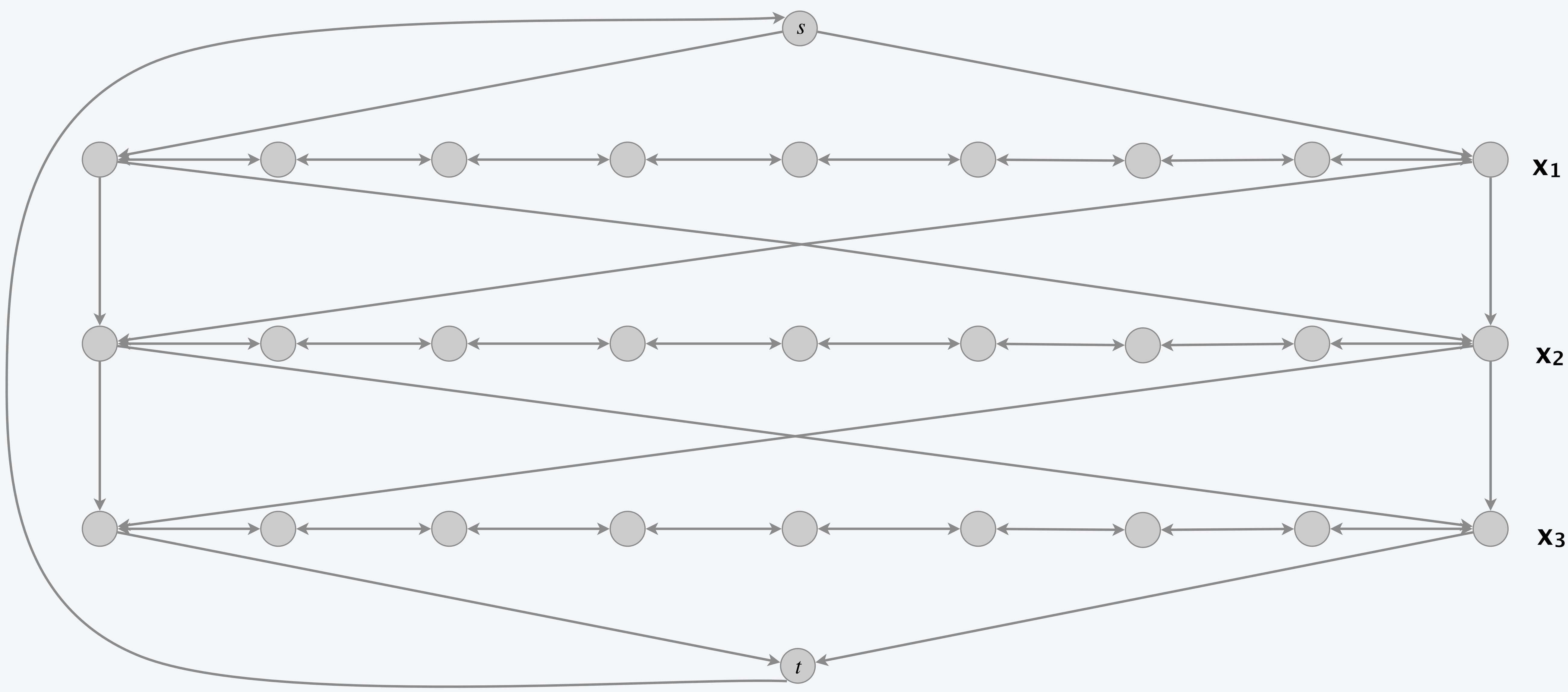
Pf. Given an instance Φ of 3-SAT, we construct an instance G of DIRECTED-HAMILTON-CYCLE that has a Hamilton cycle iff Φ is satisfiable.

Construction overview. Let n denote the number of variables in Φ. 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 Φ 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 $x_i = true$.



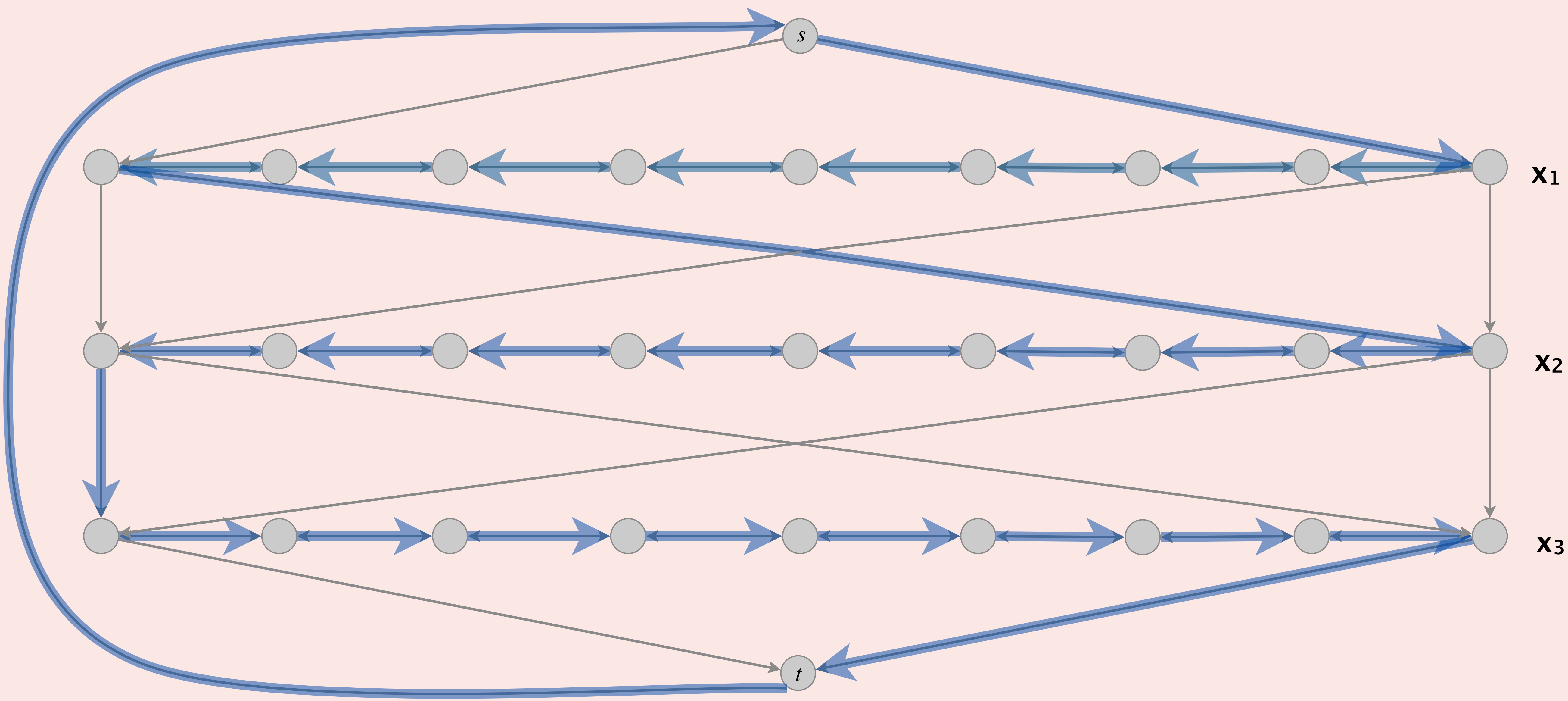
Which is truth assignment corresponding to Hamilton cycle below?

A. $x_1 = true, x_2 = true, x_3 = true$

B. $x_1 = true, x_2 = true, x_3 = false$

C. $x_1 = false, x_2 = false, x_3 = true$

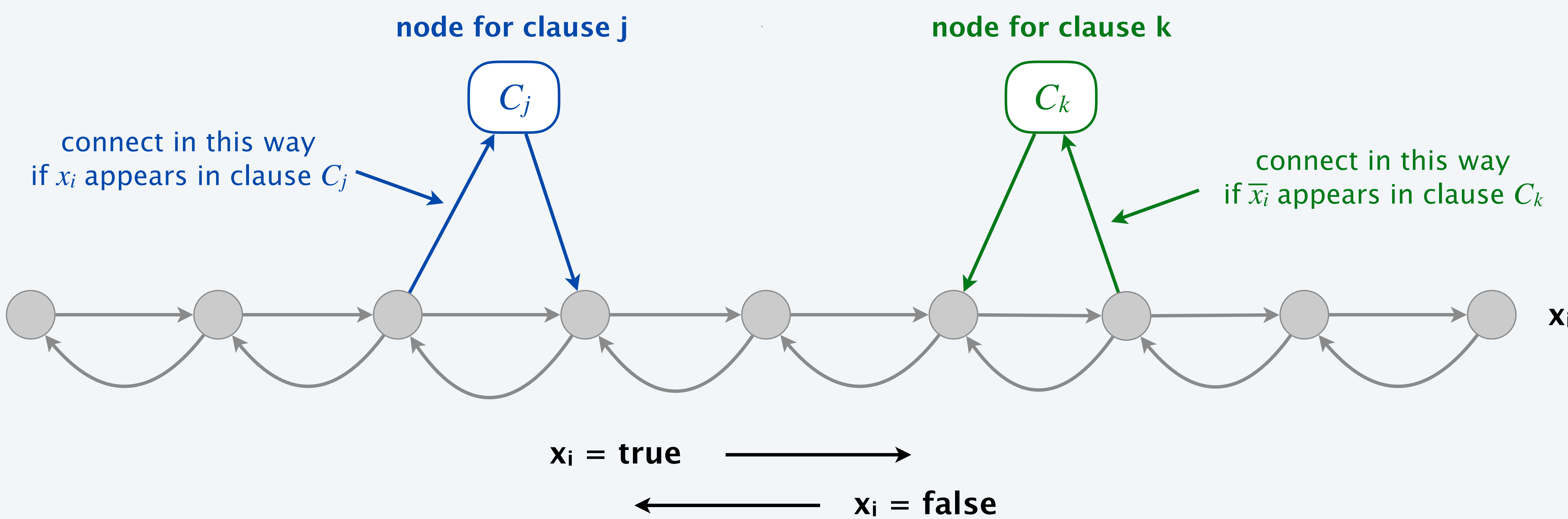
D. $x_1 = false, x_2 = false, x_3 = false$



3-satisfiability reduces to directed Hamilton cycle

Construction. Given 3-SAT instance Φ with n variables x_i and k clauses.

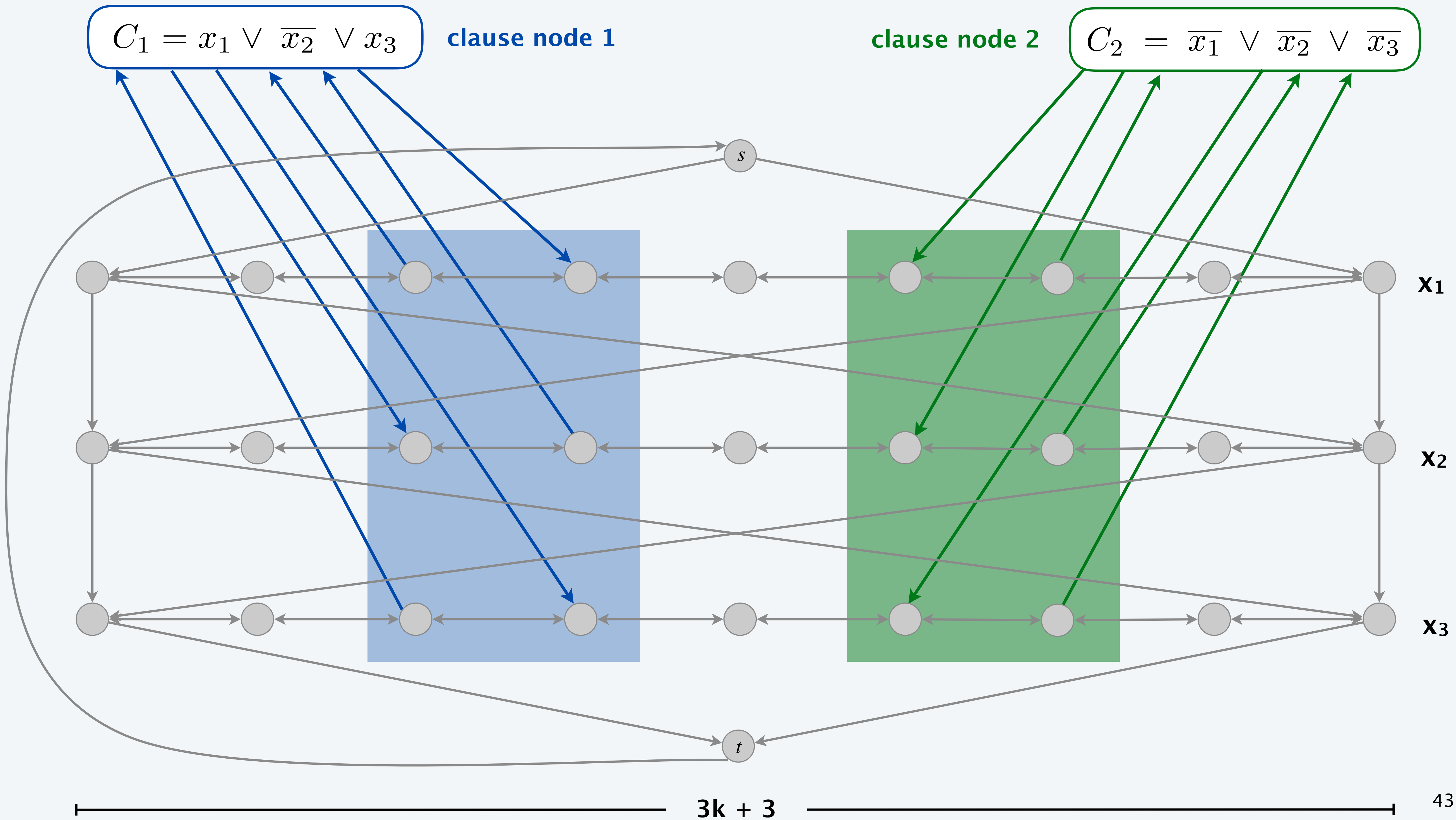
- For each clause: add a node and 2 edges per literal.



3-satisfiability reduces to directed Hamilton cycle

Construction. Given 3-SAT instance Φ 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. Φ is satisfiable iff G has a Hamilton cycle.

Pf. $\Rightarrow$

- Suppose 3-SAT instance Φ has satisfying assignment x^*.
- Then, define Hamilton cycle Γ in G as follows:
  - if $x_i^* = true$, traverse row i from left to right
  - if $x_i^* = false$, 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. Φ is satisfiable iff G has a Hamilton cycle.

Pf. $\Leftarrow$

- Suppose G has a Hamilton cycle Γ.
- If Γ 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 Γ' in $G - \{ C_1, C_2, \ldots, C_k \}$.
- Set $x_i^* = true$ if Γ' traverses row i left-to-right; otherwise, set $x_i^* = false$.
- traversed in "correct" direction, and each clause is satisfied. ▪

3-satisfiability reduces to longest path

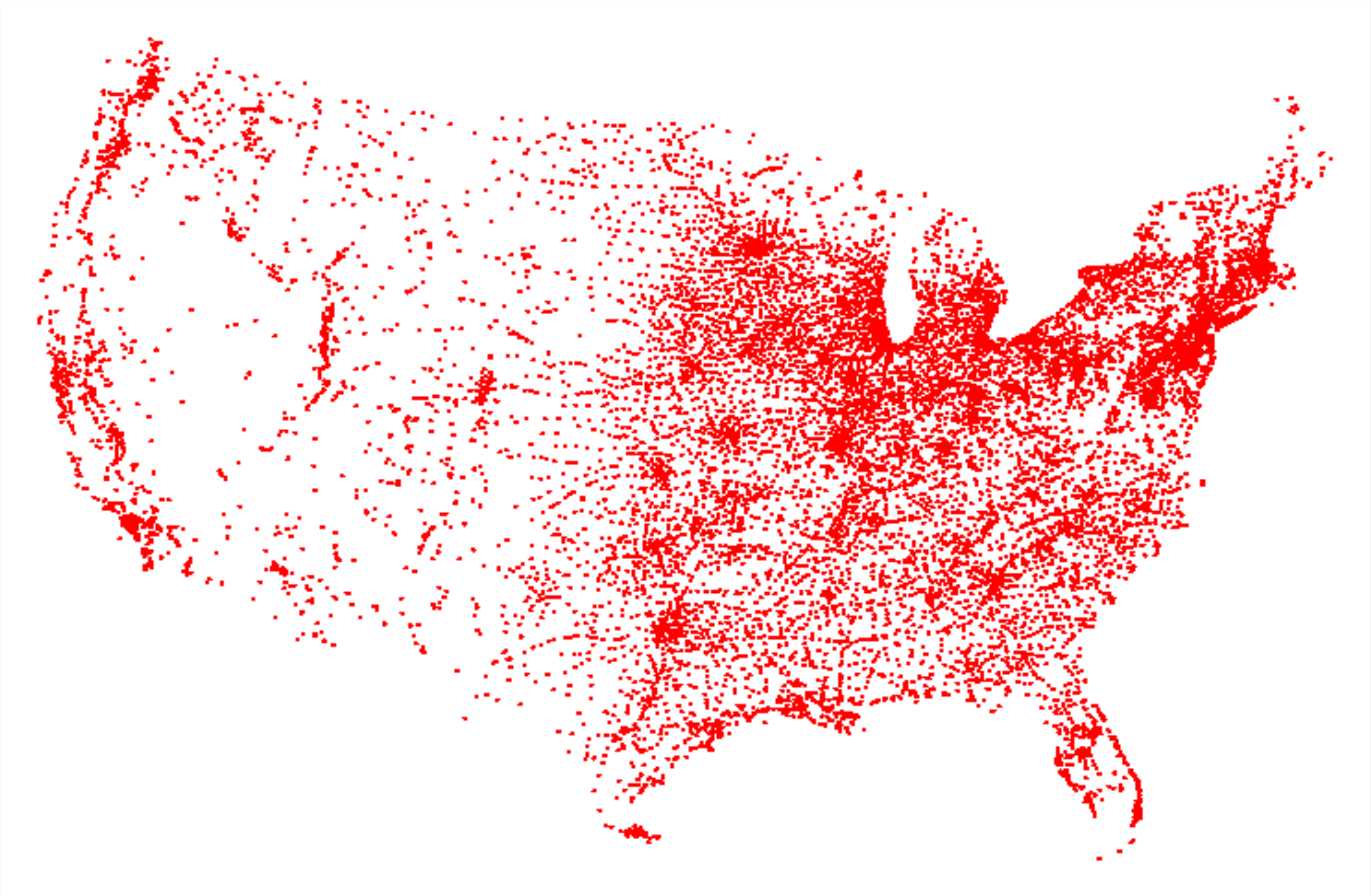
LONGEST-PATH. Given a directed graph $G = (V, E)$, does there exists a simple path consisting of at least k edges?

Theorem. 3-SAT $\leq_P$ LONGEST-PATH.

Pf 1. Redo proof for DIR-HAM-CYCLE, ignoring back-edge from t to s.
Pf 2. Show HAM-CYCLE $\leq_P$ LONGEST-PATH.

Traveling salesperson problem

TSP. Given a set of n cities and a pairwise distance function $d(u, v)$, is there a tour of length $\leq D$?



13,509 cities in the United States
http://www.tsp.gatech.edu

Traveling salesperson problem

TSP. Given a set of n cities and a pairwise distance function $d(u, v)$, is there a tour of length $\leq D$?

optimal TSP tour
http://www.tsp.gatech.edu

Hamilton cycle reduces to traveling salesperson problem

TSP. Given a set of n cities and a pairwise distance function $d(u, v)$, is there a tour of length $\leq D$?

HAM-CYCLE. Given an undirected graph $G = (V, E)$, does there exist a simple cycle Γ that contains every node in V ?

Theorem. HAM-CYCLE $\leq_P$ TSP.

Pf.

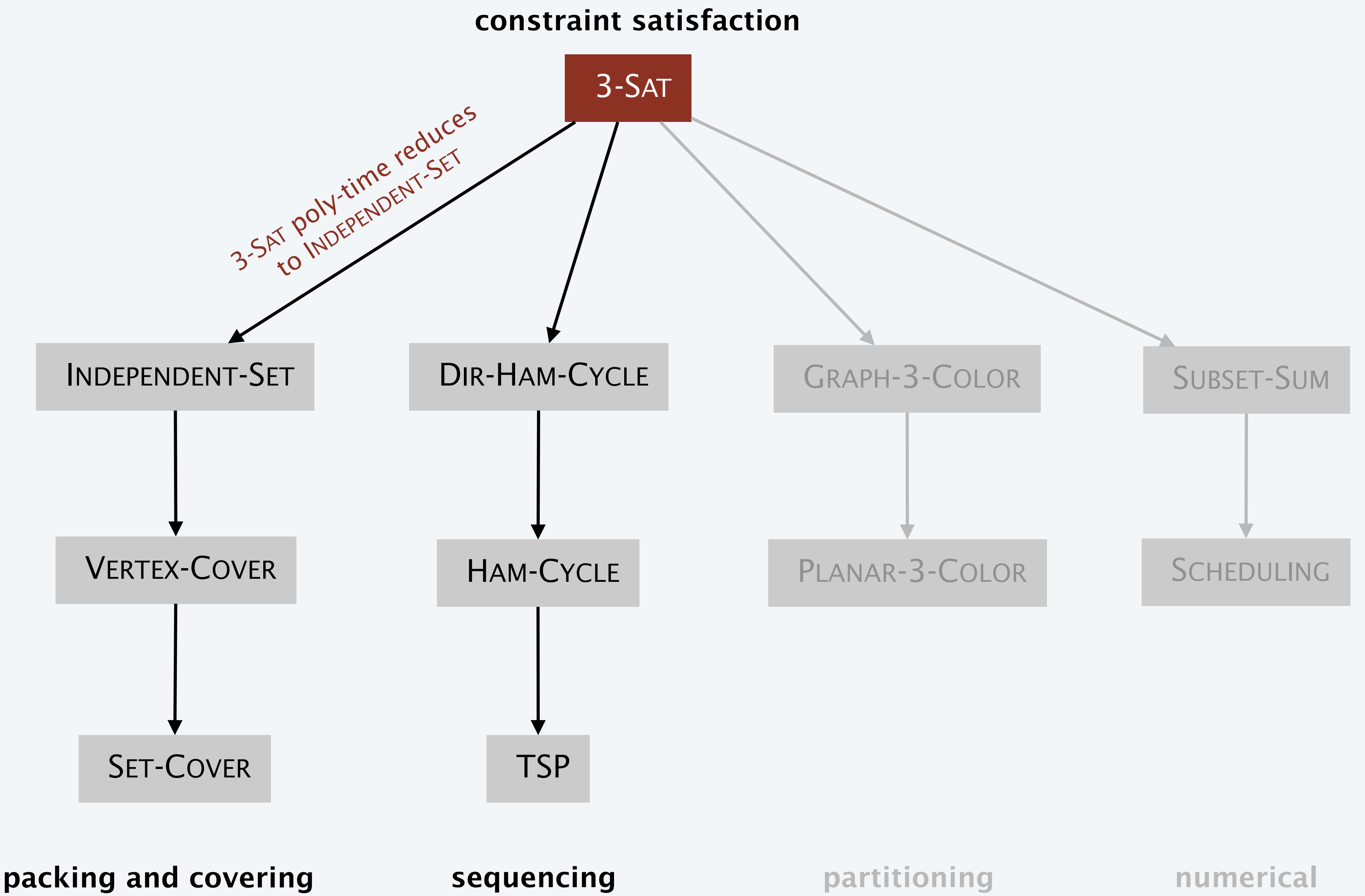
- Given instance $G = (V, E)$ of HAM-CYCLE, create n cities with distance function

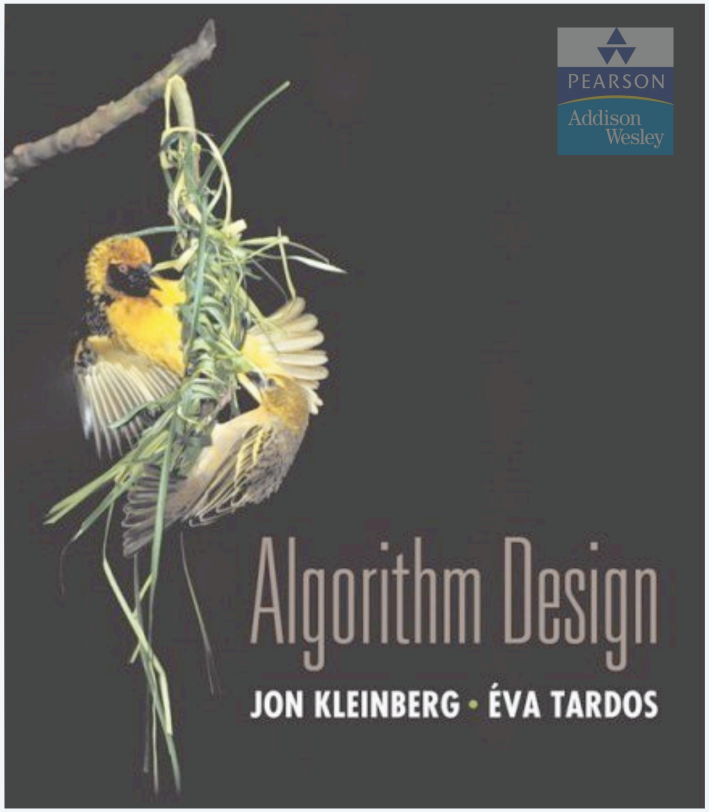
$$d(u, v) = \begin{cases} 1 & \text{if } (u, v) \in E \\ 2 & \text{if } (u, v) \notin E \end{cases}$$

- TSP instance has tour of length $\leq n$ iff G has a Hamilton cycle. ▪

Remark. TSP instance satisfies triangle inequality: $d(u, w) \leq d(u, v) + d(v, w)$.

Polynomial-time reductions





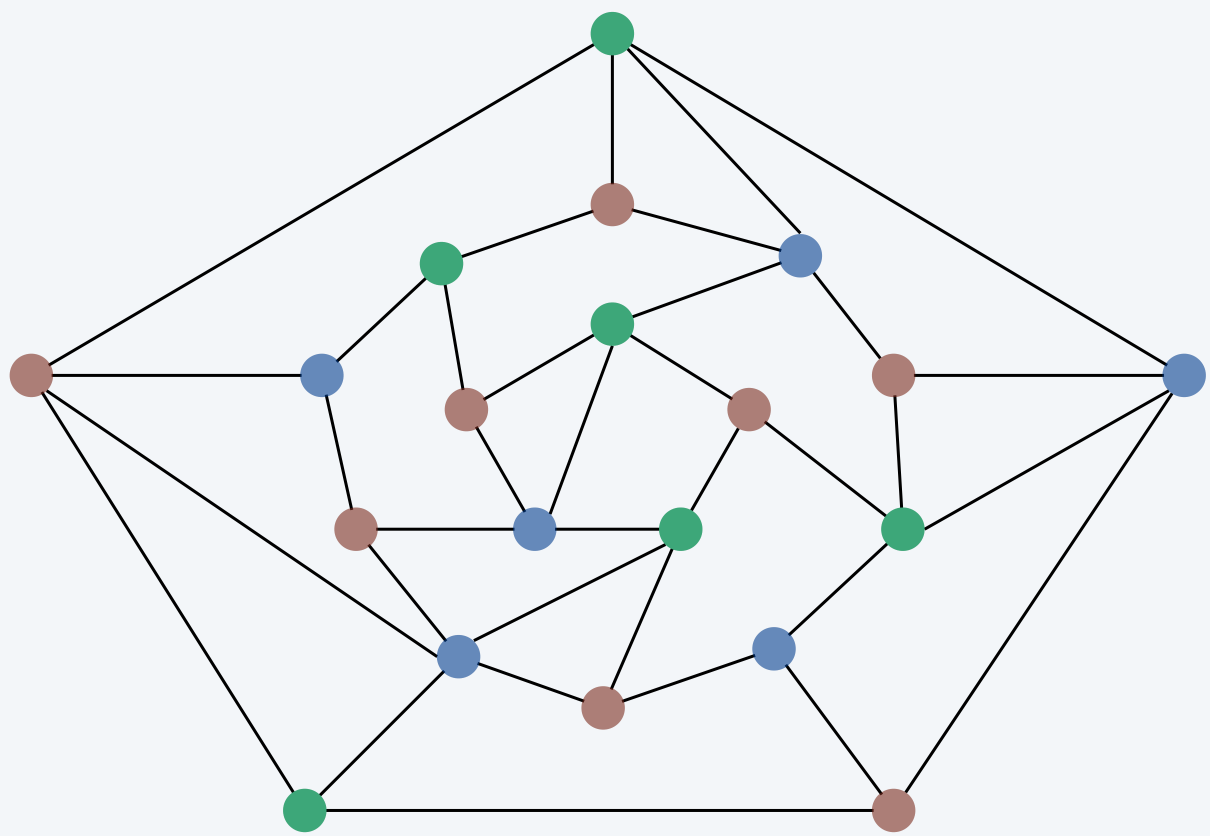
SECTION 8.7

8. INTRACTABILITY I

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

3-colorability

3-COLOR. Given an undirected graph G, can the nodes be colored red, green, and blue so that no adjacent nodes have the same color?



yes instance

Application: register allocation

Register allocation. Assign program variables to machine register so that no more than k registers are used and no two program variables that are needed at the same time are assigned to the same register.

Interference graph. Nodes are program variables names; edge between u and v if there exists an operation where both u and v are "live" at the same time.

Observation. [Chaitin 1982] Can solve register allocation problem iff interference graph is k-colorable.

Fact. 3-COLOR $\le_P$ K-REGISTER-ALLOCATION for any constant $k \ge 3$.

REGISTER ALLOCATION & SPILLING VIA GRAPH COLORING

G. J. Chaitin
IBM Research
P.O.Box 218, Yorktown Heights, NY 10598

3-satisfiability reduces to 3-colorability

Theorem. 3-SAT $\le_P$ 3-COLOR.

Pf. Given 3-SAT instance Φ, we construct an instance of 3-COLOR that is 3-colorable iff Φ is satisfiable.

3-satisfiability reduces to 3-colorability

Construction.

(i) Create a graph G with a node for each literal.

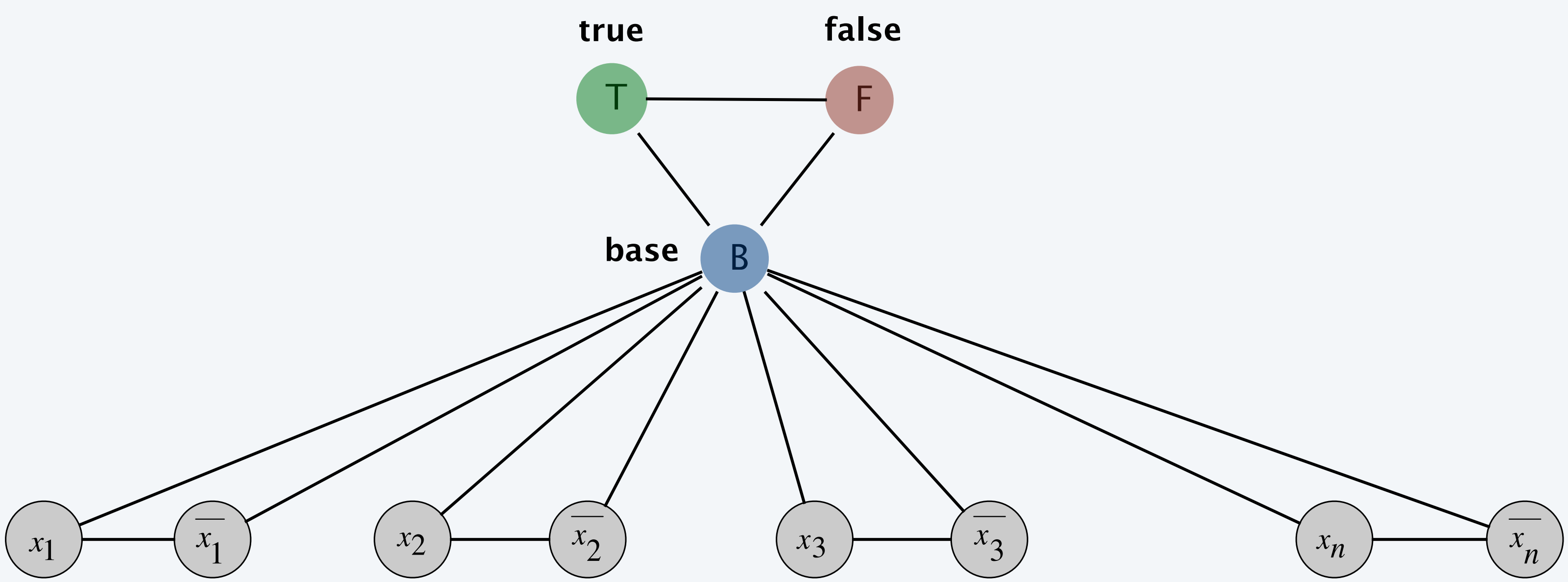
(ii) Connect each literal to its negation.

(iii) Create 3 new nodes T, F, and B; connect them in a triangle.

(iv) Connect each literal to B.

(v) For each clause C_j, add a gadget of 6 nodes and 13 edges.

to be described later

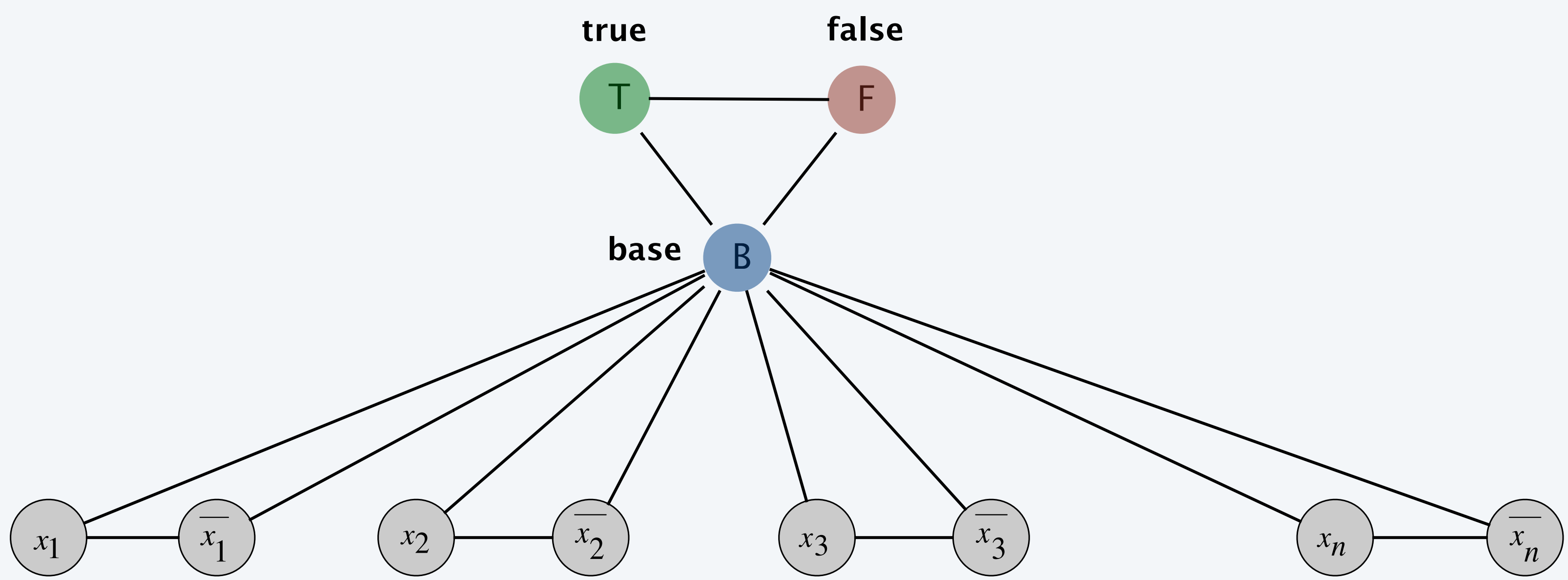


3-satisfiability reduces to 3-colorability

Lemma. Graph G is 3-colorable iff Φ is satisfiable.

Pf. $\Rightarrow$ Suppose graph G is 3-colorable.

- Consider assignment that sets all T literals to true.
- (iv) ensures each literal is T or F.
- (ii) ensures a literal and its negation are opposites.

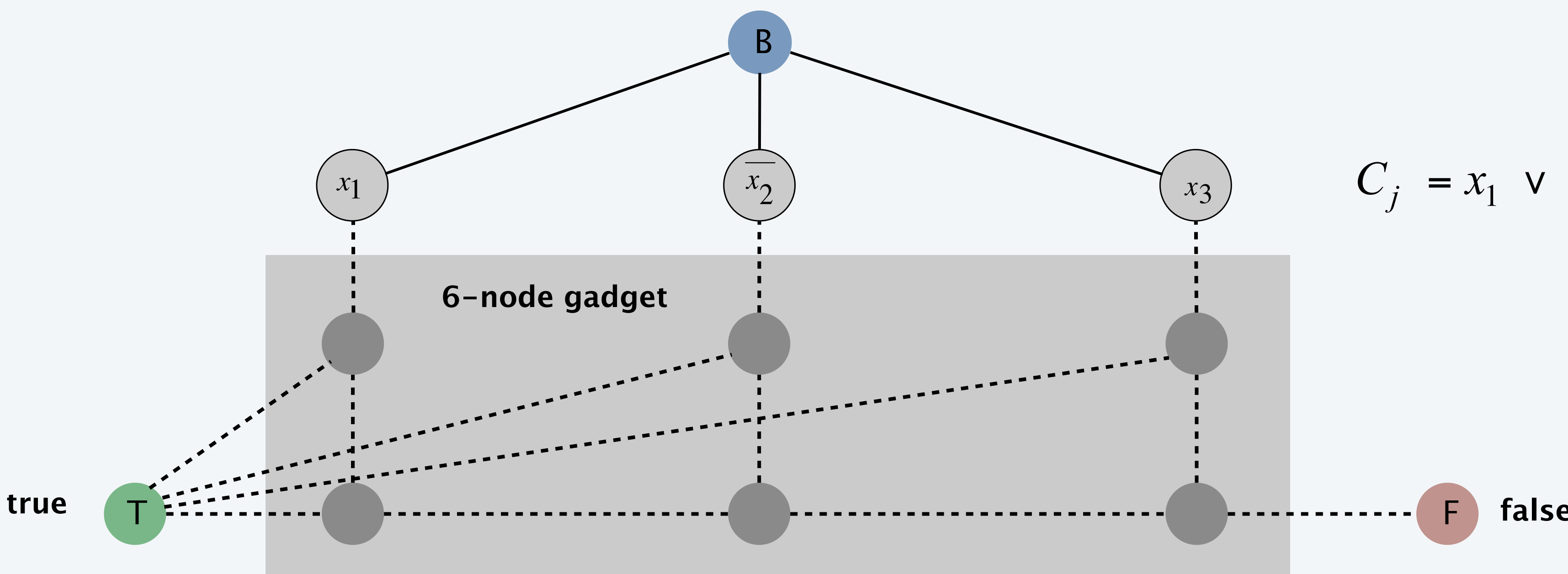


3-satisfiability reduces to 3-colorability

Lemma. Graph G is 3-colorable iff Φ is satisfiable.

Pf. $\Rightarrow$ Suppose graph G is 3-colorable.

- Consider assignment that sets all T literals to true.
- (iv) ensures each literal is T or F.
- (ii) ensures a literal and its negation are opposites.
- (v) ensures at least one literal in each clause is T.



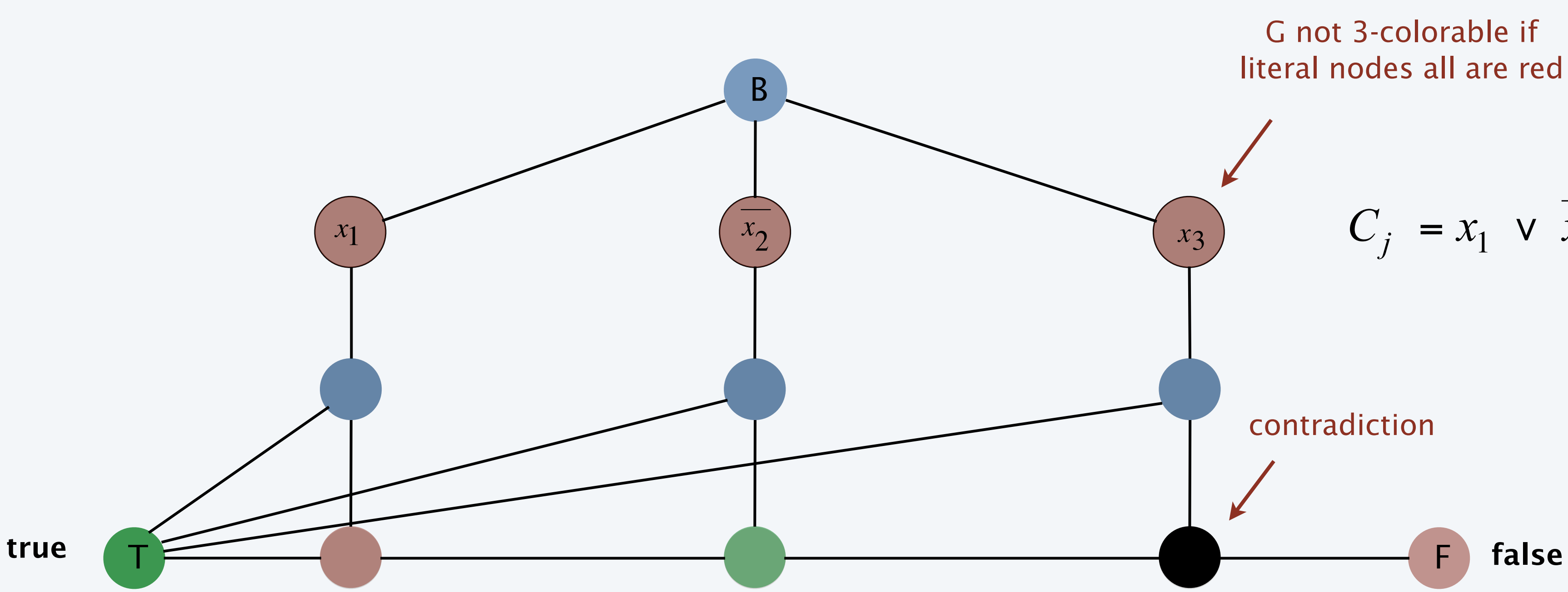
$$C_j = x_1 \lor \overline{x_2} \lor x_3$$

3-satisfiability reduces to 3-colorability

Lemma. Graph G is 3-colorable iff Φ is satisfiable.

Pf. $\Rightarrow$ Suppose graph G is 3-colorable.

- Consider assignment that sets all T literals to true.
- (iv) ensures each literal is T or F.
- (ii) ensures a literal and its negation are opposites.
- (v) ensures at least one literal in each clause is T.



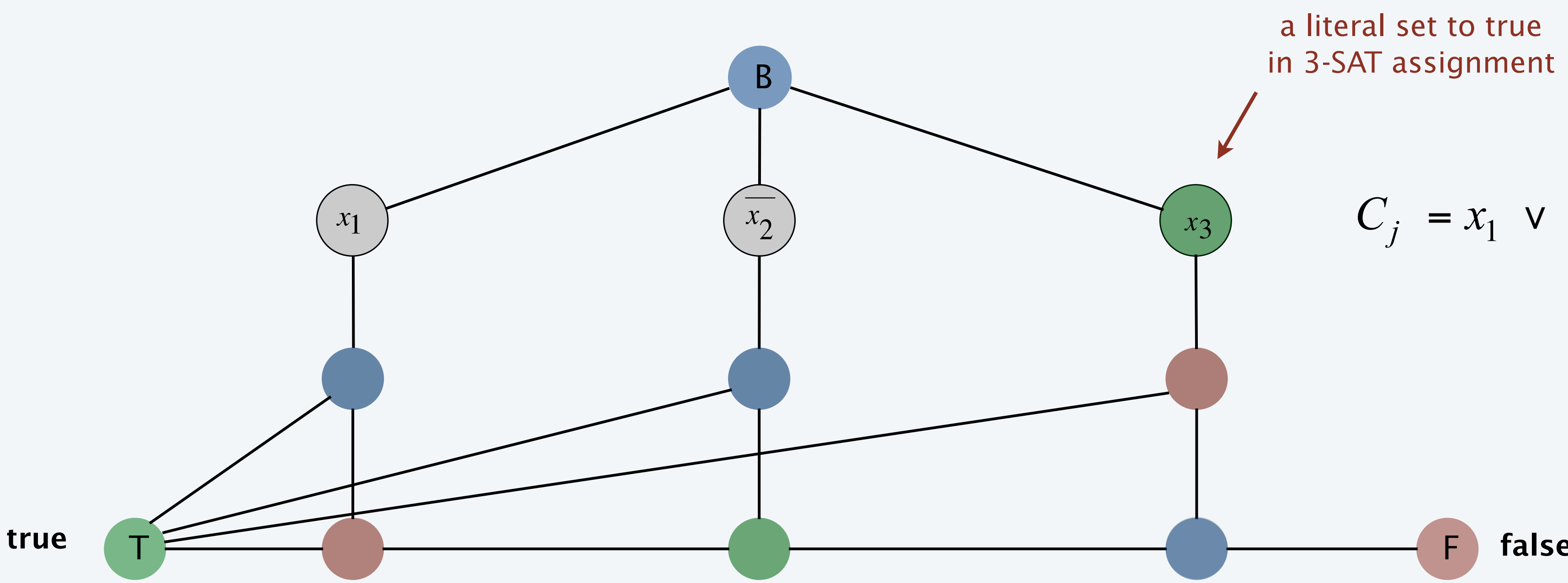
$$C_j = x_1 \vee \overline{x_2} \vee x_3$$

3-satisfiability reduces to 3-colorability

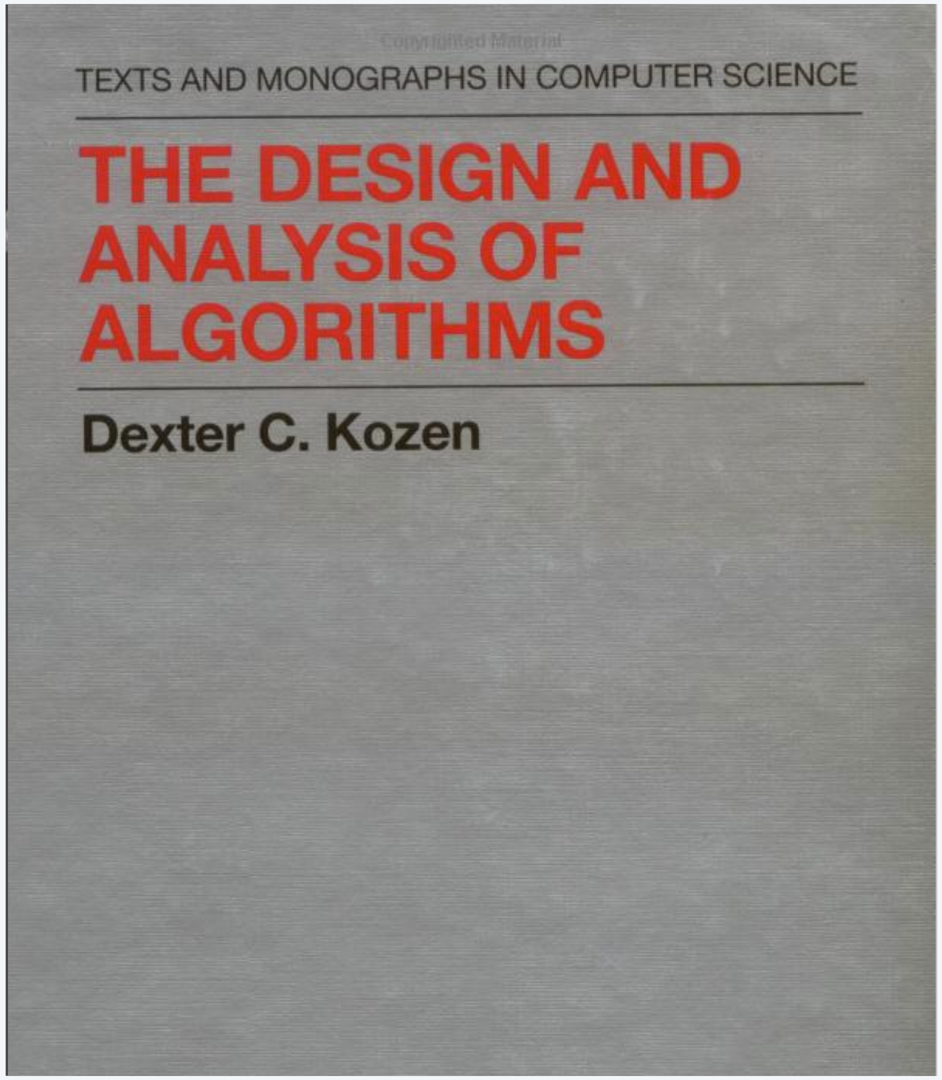
Lemma. Graph G is 3-colorable iff Φ is satisfiable.

Pf. $\Leftarrow$ Suppose 3-SAT instance Φ is satisfiable.

- Color all true literals T.
- Color node below green node F, and node below that B.
- Color remaining middle row nodes B.
- Color remaining bottom nodes T or F as forced. ▪



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



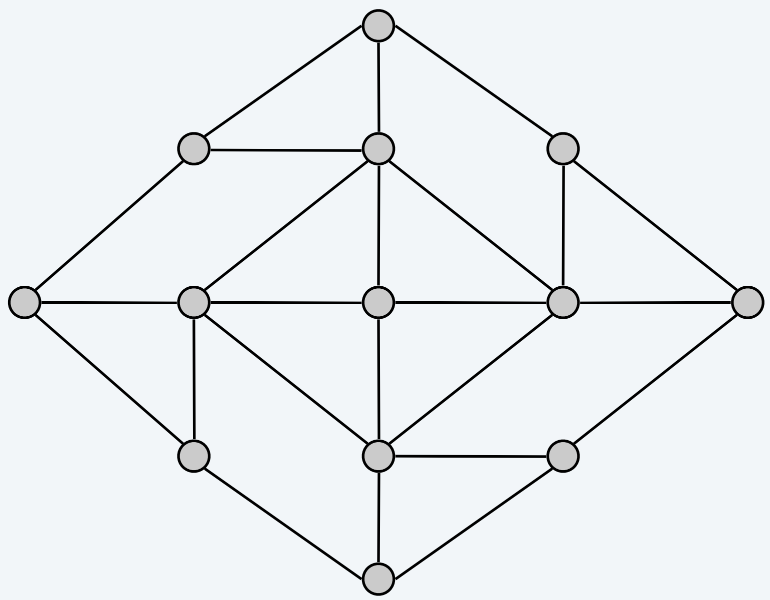
SECTION 23.1

INTRACTABILITY III

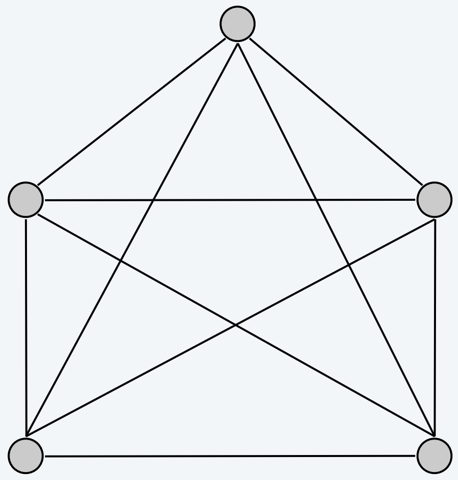
- *special cases: trees*
- *special cases: planarity*
- *approximation algorithms: vertex cover*
- *approximation algorithms: knapsack*
- *exponential algorithms: 3-SAT*
- *exponential algorithms: TSP*

Planarity

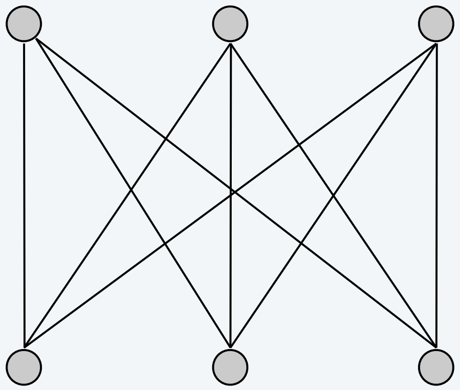
Def. A graph is planar if it can be embedded in the plane in such a way that no two edges cross.



planar



K_5 is nonplanar



$K_{3,3}$ is nonplanar

Applications. VLSI circuit design, computer graphics, ...

Planarity testing

Theorem. [Hopcroft–Tarjan 1974] There exists an $O(n)$ time algorithm to determine whether a graph is planar.

simple planar graph
has at $\leq 3n$ edges

Efficient Planarity Testing

JOHN HOPCROFT AND ROBERT TARJAN

Cornell University, Ithaca, New York

ABSTRACT. This paper describes an efficient algorithm to determine whether an arbitrary graph G can be embedded in the plane. The algorithm may be viewed as an iterative version of a method originally proposed by Auslander and Parter and correctly formulated by Goldstein. The algorithm uses depth-first search and has $O(V)$ time and space bounds, where V is the number of vertices in G. An ALGOL implementation of the algorithm successfully tested graphs with as many as 900 vertices in less than 12 seconds.

Problems on planar graphs

Fact 0. Many graph problems can be solved faster in planar graphs.

Ex. Shortest paths, max flow, MST, matchings, ...

Fact 1. Some **NP**-complete problems become tractable in planar graphs.

Ex. MAX-CUT, ISING, CLIQUE, GRAPH-ISOMORPHISM, 4-COLOR, ...

Fact 2. Other **NP**-complete problems become easier in planar graphs.

Ex. INDEPENDENT-SET, VERTEX-COVER, TSP, STEINER-TREE, ...

An $O(n \log n)$ Algorithm for Maximum *st*-Flow in a Directed Planar Graph

GLENCORA BORRADAILE AND PHILIP KLEIN

Brown University, Providence, Rhode Island

Abstract. We give the first correct $O(n \log n)$ algorithm for finding a maximum st-flow in a directed planar graph. After a preprocessing step that consists in finding single-source shortest-path distances in the dual, the algorithm consists of repeatedly saturating the leftmost residual s-to-t path.

SIAM J. COMPUT.
Vol. 9, No. 3, August 1980

0097-5397/80/0903-0013 $01.00/0

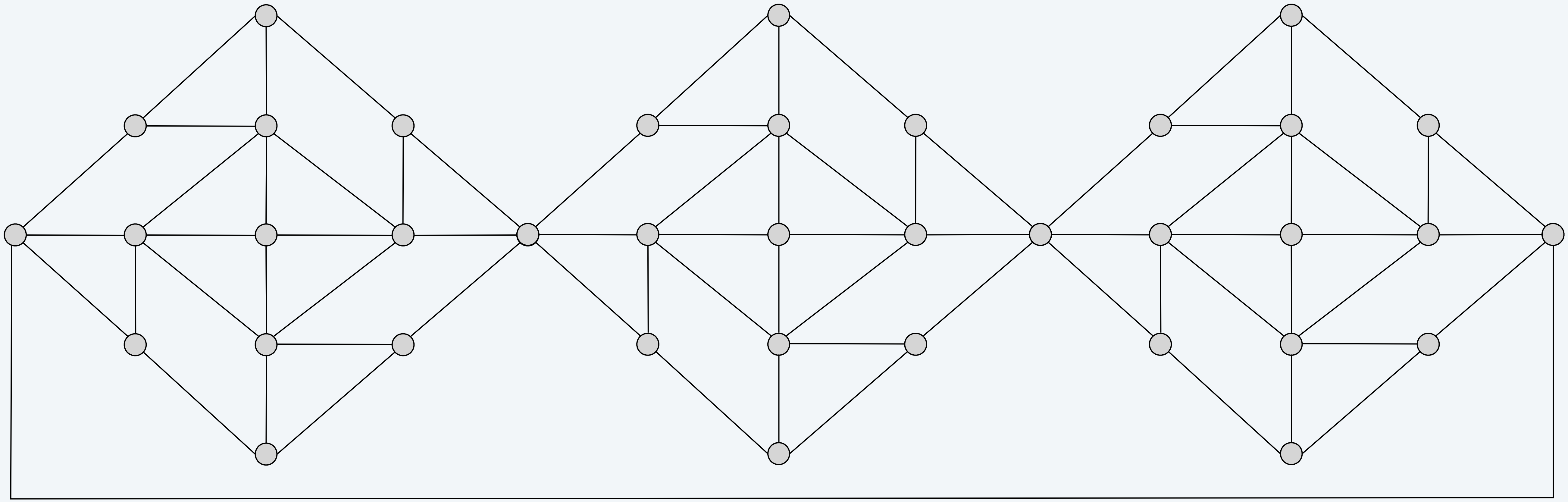
APPLICATIONS OF A PLANAR SEPARATOR THEOREM*

RICHARD J. LIPTON† AND ROBERT ENDRE TARJAN‡

Abstract. Any n-vertex planar graph has the property that it can be divided into components of roughly equal size by removing only $O(\sqrt{n})$ vertices. This separator theorem, in combination with a divide-and-conquer strategy, leads to many new complexity results for planar graph problems. This paper describes some of these results.

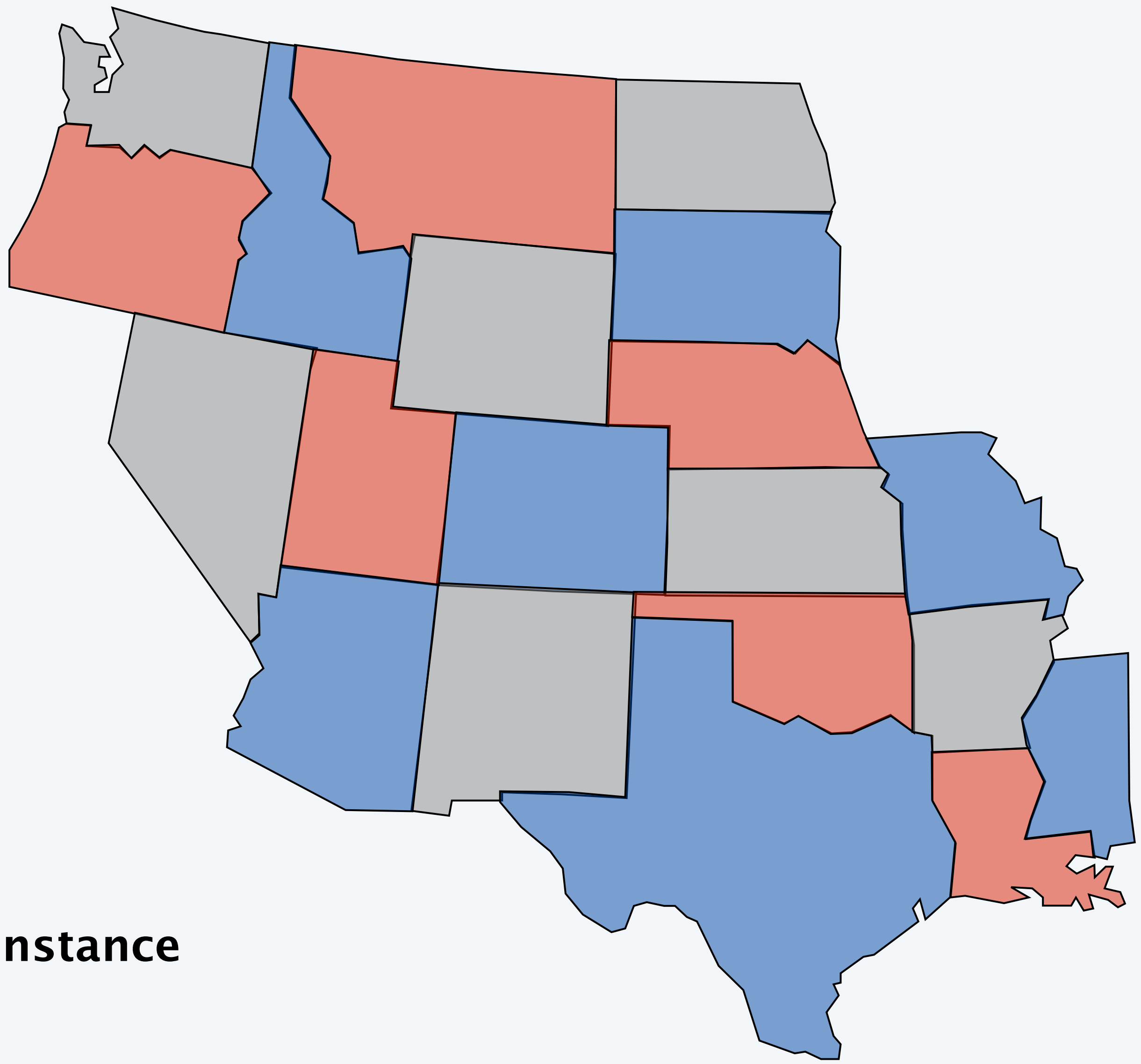
Planar graph 3-colorability

PLANAR-3-COLOR. Given a planar graph, can it be colored using 3 colors so that no two adjacent nodes have the same color?



Planar map 3-colorability

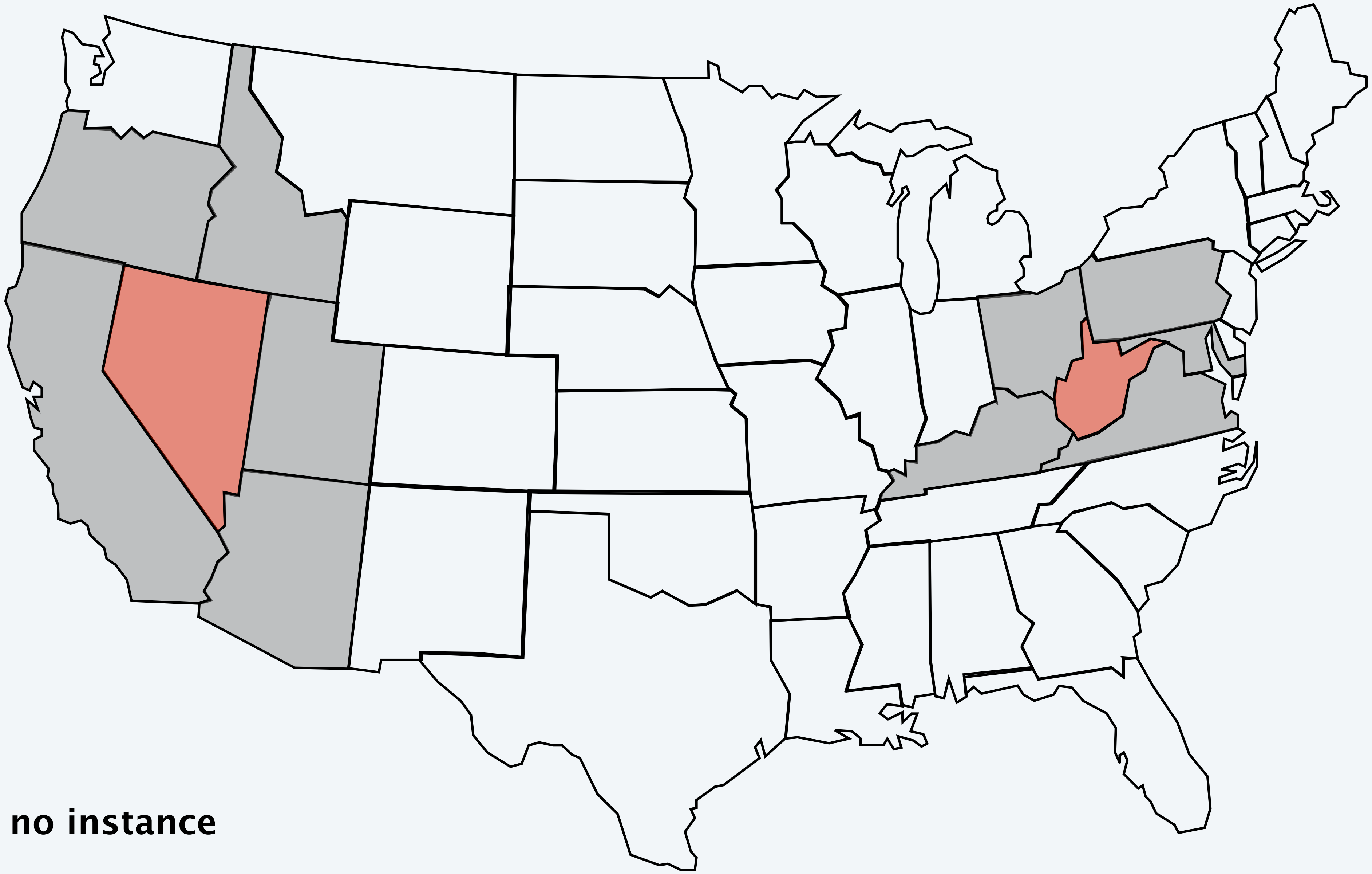
PLANAR-MAP-3-COLOR. Given a planar map, can it be colored using 3 colors so that no two adjacent regions have the same color?



yes instance

Planar map 3-colorability

PLANAR-MAP-3-COLOR. Given a planar map, can it be colored using 3 colors so that no two adjacent regions have the same color?



no instance

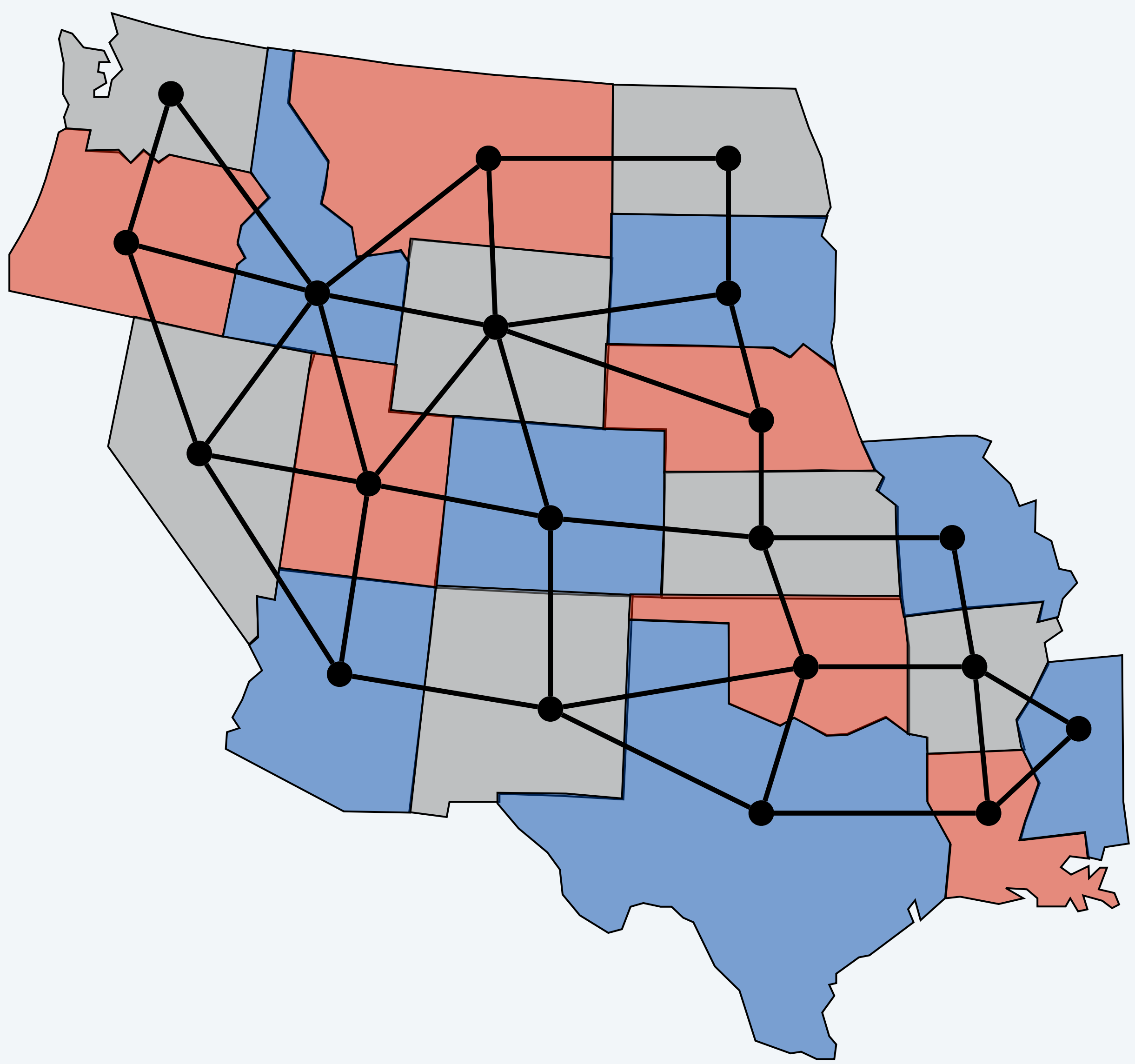
Planar graph and map 3-colorability reduce to one another

Theorem. PLANAR-3-COLOR $\equiv_P$ PLANAR-MAP-3-COLOR.

Pf sketch.

- Nodes correspond to regions.
- Two nodes are adjacent iff they share a nontrivial border.

e.g., not Arizona and Colorado



Planar 3-colorability is NP-complete

Theorem. PLANAR-3-COLOR $\in$ **NP**-complete.

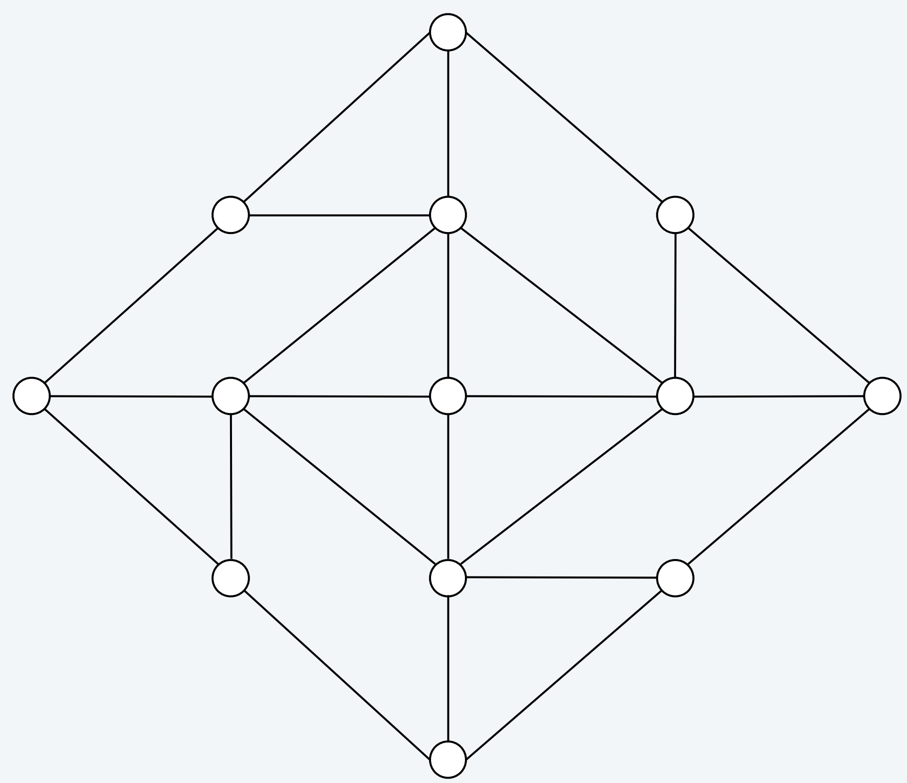
Pf.

- Easy to see that PLANAR-3-COLOR $\in$ **NP**.
- We show 3-COLOR $\leq_P$ PLANAR-3-COLOR.
- Given 3-COLOR instance G, we construct an instance of PLANAR-3-COLOR that is 3-colorable iff G is 3-colorable.

Planar 3-colorability is NP-complete

Lemma. W is a planar graph such that:

- In any 3-coloring of W, opposite corners have the same color.
- Any assignment of colors to the corners in which opposite corners have the same color extends to a 3-coloring of W.



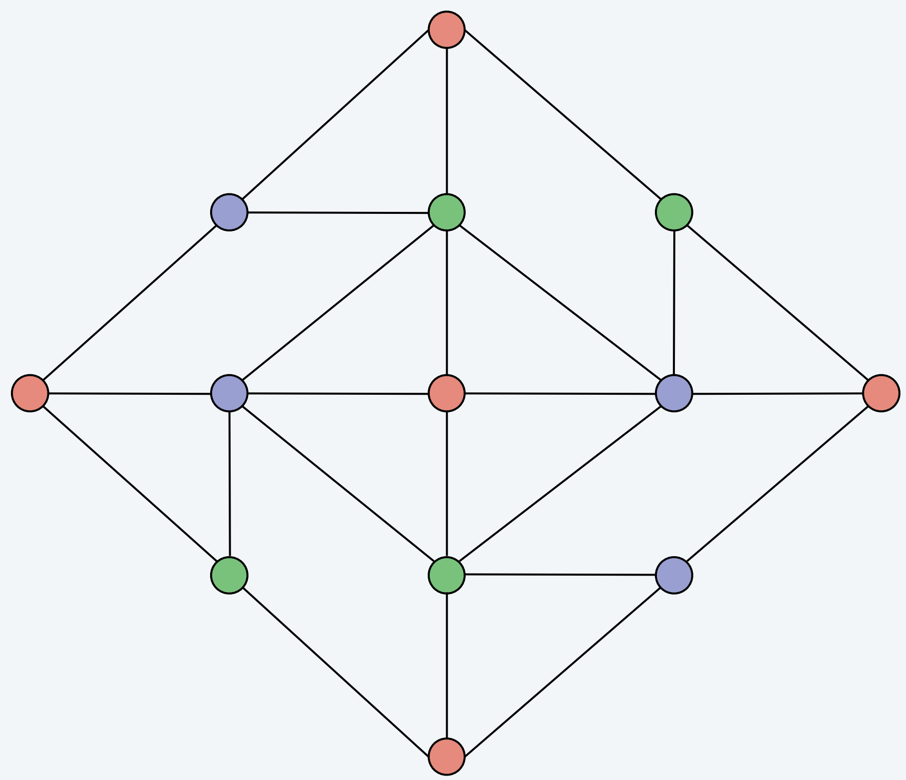
planar gadget W

Planar 3-colorability is NP-complete

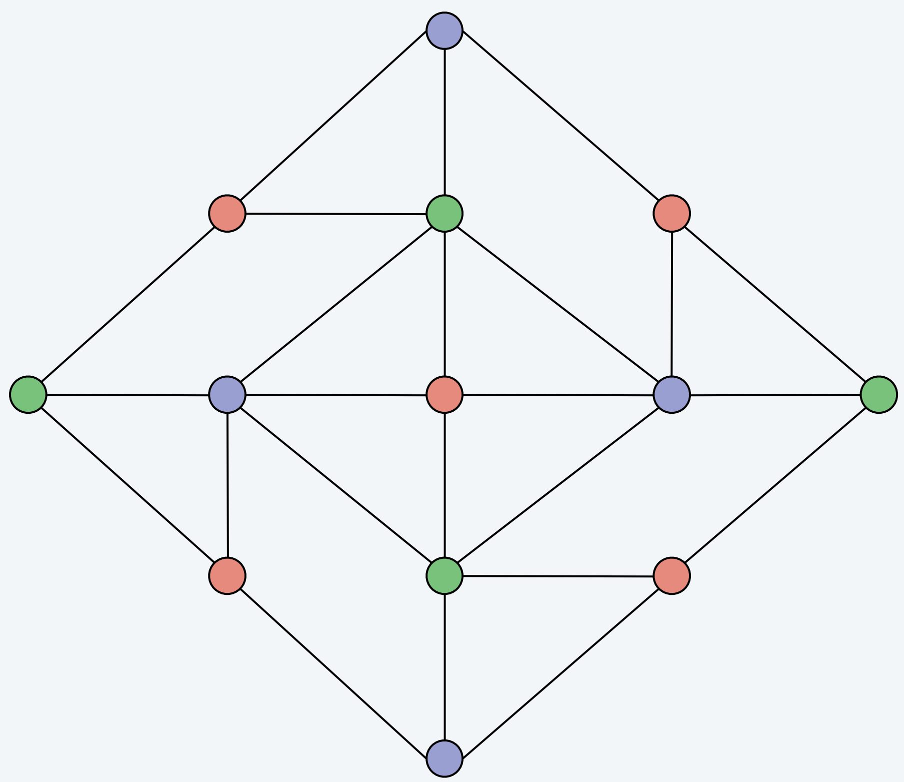
Lemma. W is a planar graph such that:

- In any 3-coloring of W, opposite corners have the same color.
- Any assignment of colors to the corners in which opposite corners have the same color extends to a 3-coloring of W.

Pf. The only 3-colorings (modulo permutations) of W are shown below. ▪



planar gadget W

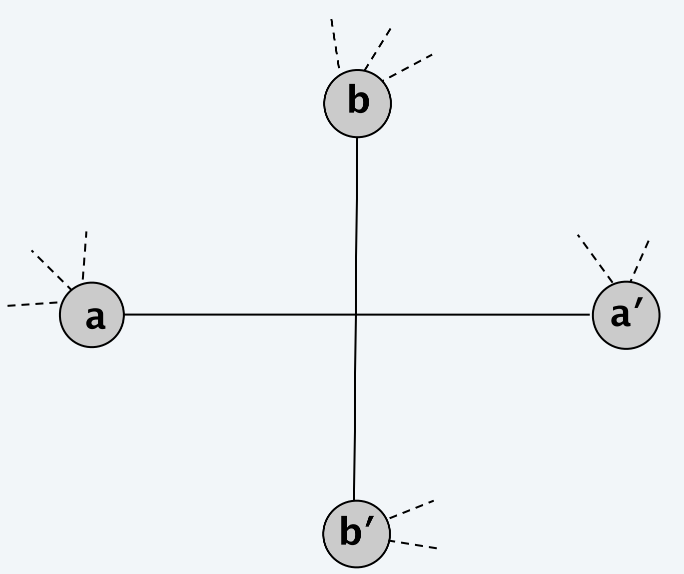


Planar 3-colorability is NP-complete

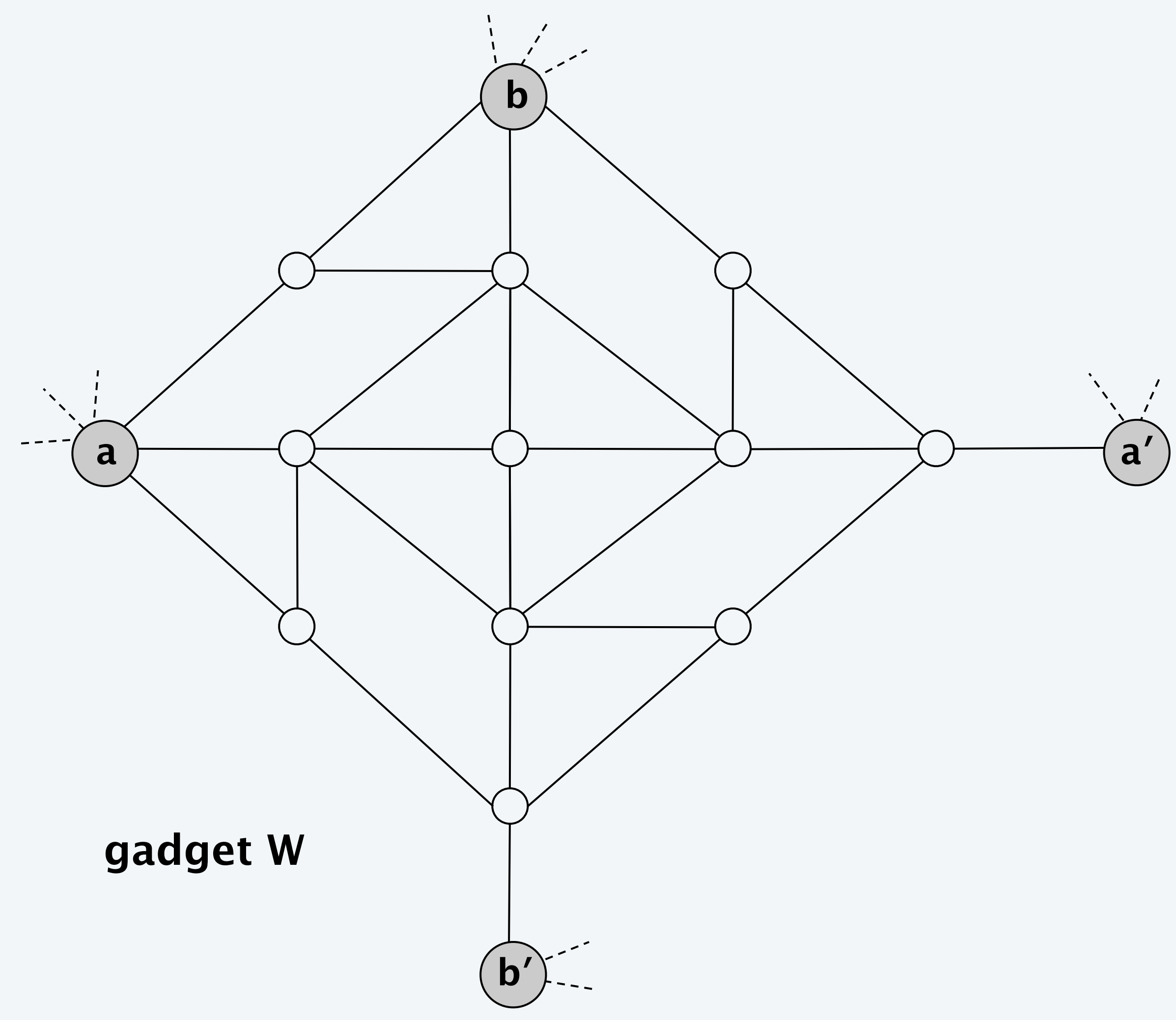
Construction. Given instance G of 3-COLOR, draw G in plane, letting edges cross. Form planar G' by replacing each edge crossing with planar gadget W.

Lemma. G is 3-colorable iff G' is 3-colorable.

- In any 3-coloring of W, $a \neq a'$ and $b \neq b'$.
- If $a \neq a'$ and $b \neq b'$ then can extend to a 3-coloring of W.



a crossing



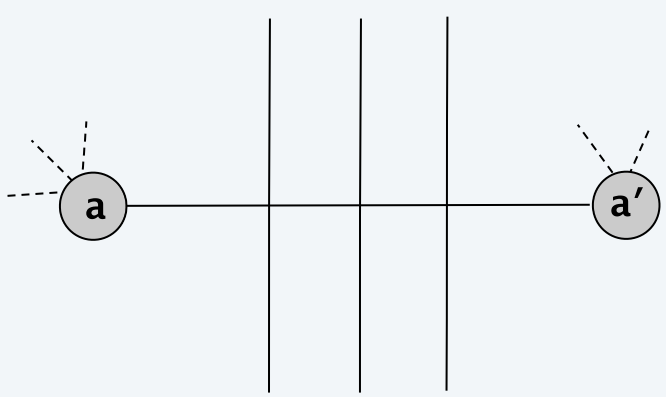
gadget W

Planar 3-colorability is NP-complete

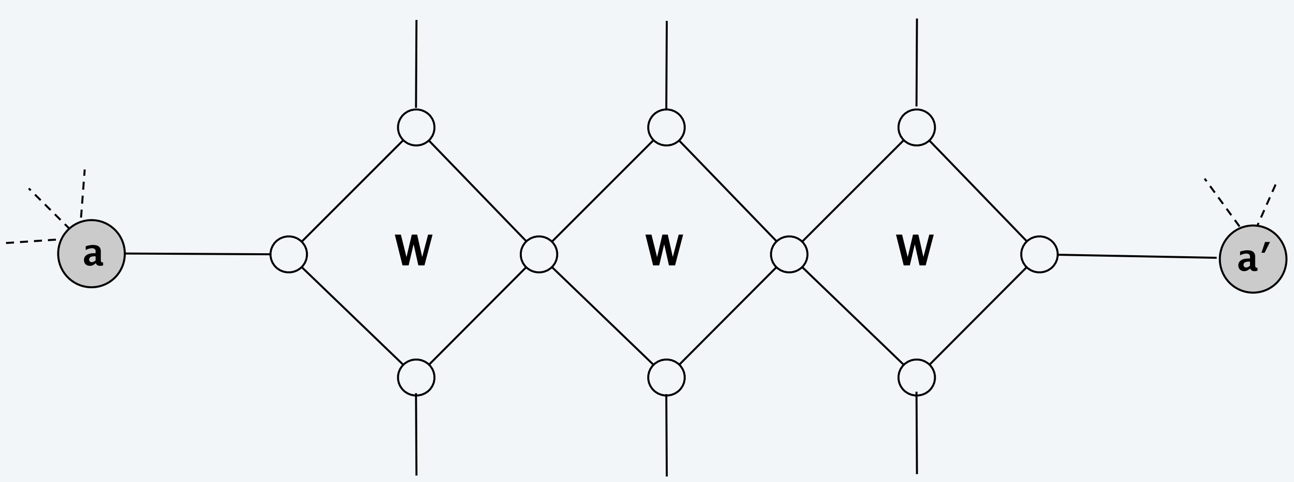
Construction. Given instance G of 3-COLOR, draw G in plane, letting edges cross. Form planar G' by replacing each edge crossing with planar gadget W.

Lemma. G is 3-colorable iff G' is 3-colorable.

- In any 3-coloring of W, $a \neq a'$ and $b \neq b'$.
- If $a \neq a'$ and $b \neq b'$ then can extend to a 3-coloring of W.



multiple crossings

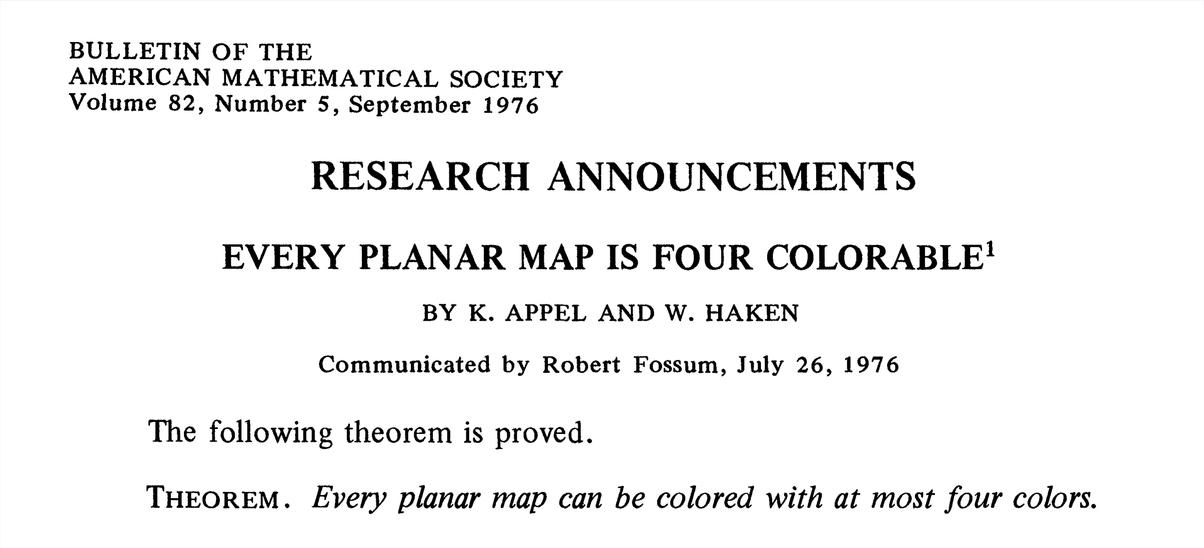


concatenate copies of gadget W

Planar map k-colorability

Theorem. [Appel–Haken 1976] Every planar map is 4-colorable.

- Resolved century-old open problem.
- Used 50 days of computer time to deal with many special cases.
- First major theorem to be proved using computer.



BULLETIN OF THE
AMERICAN MATHEMATICAL SOCIETY
Volume 82, Number 5, September 1976

RESEARCH ANNOUNCEMENTS

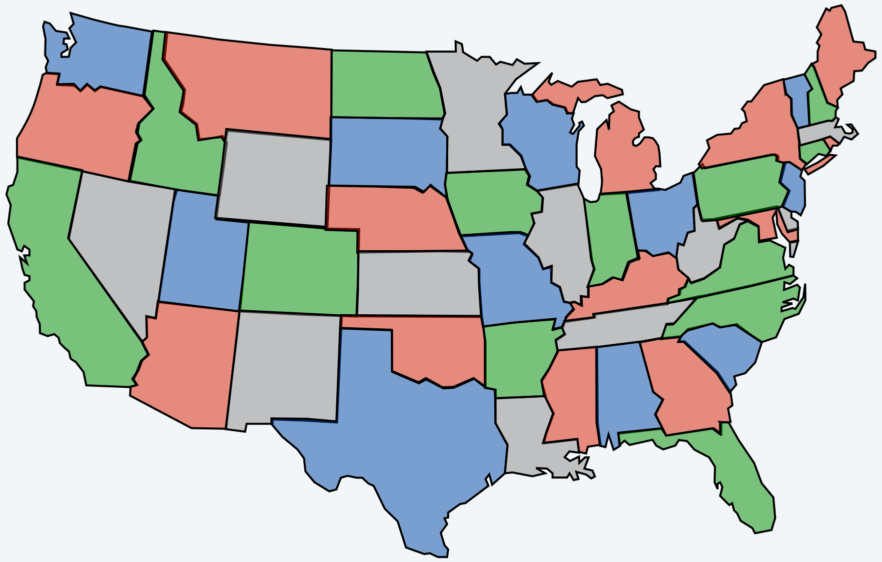
EVERY PLANAR MAP IS FOUR COLORABLE[1]

BY K. APPEL AND W. HAKEN

Communicated by Robert Fossum, July 26, 1976

The following theorem is proved.

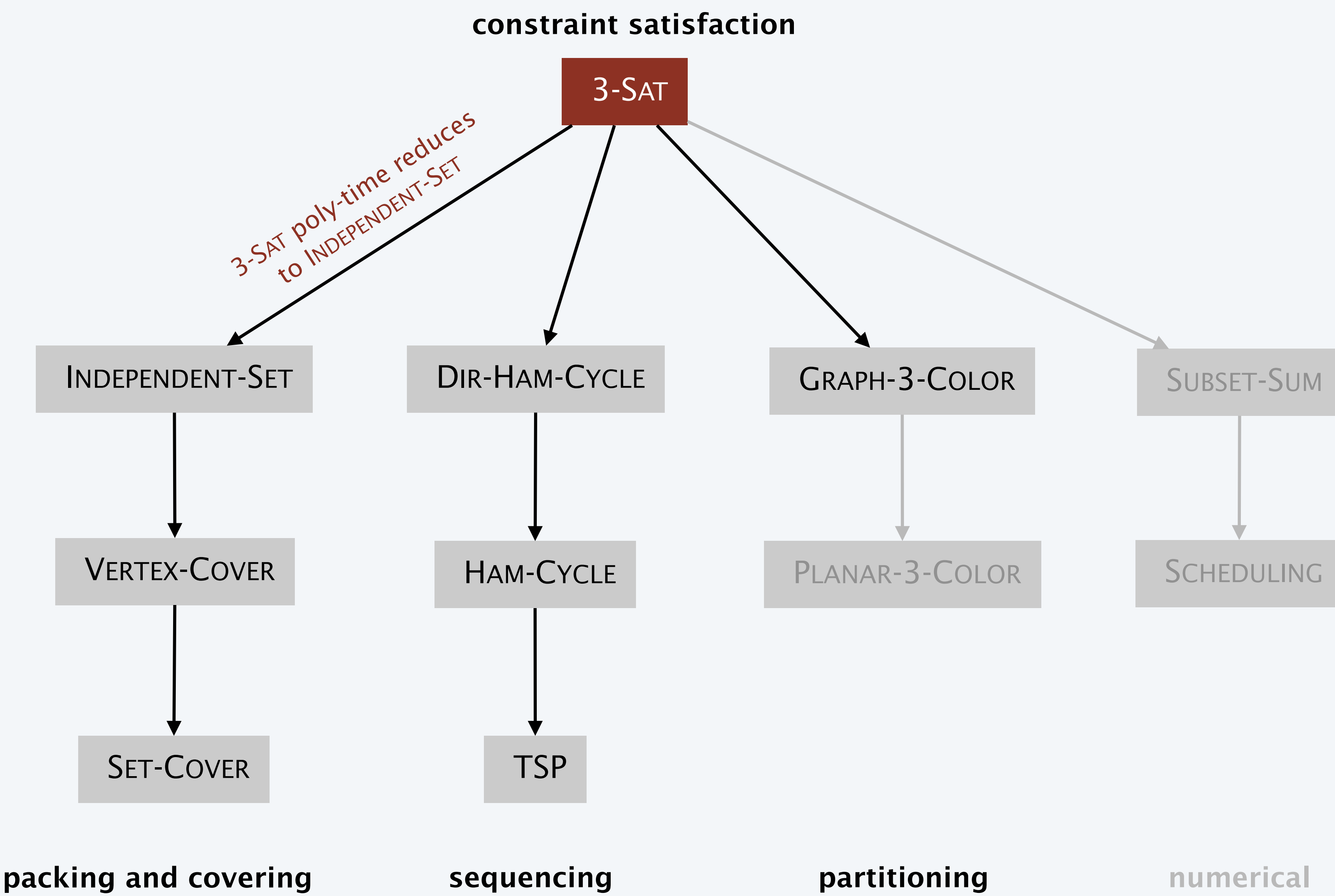
THEOREM. *Every planar map can be colored with at most four colors.*

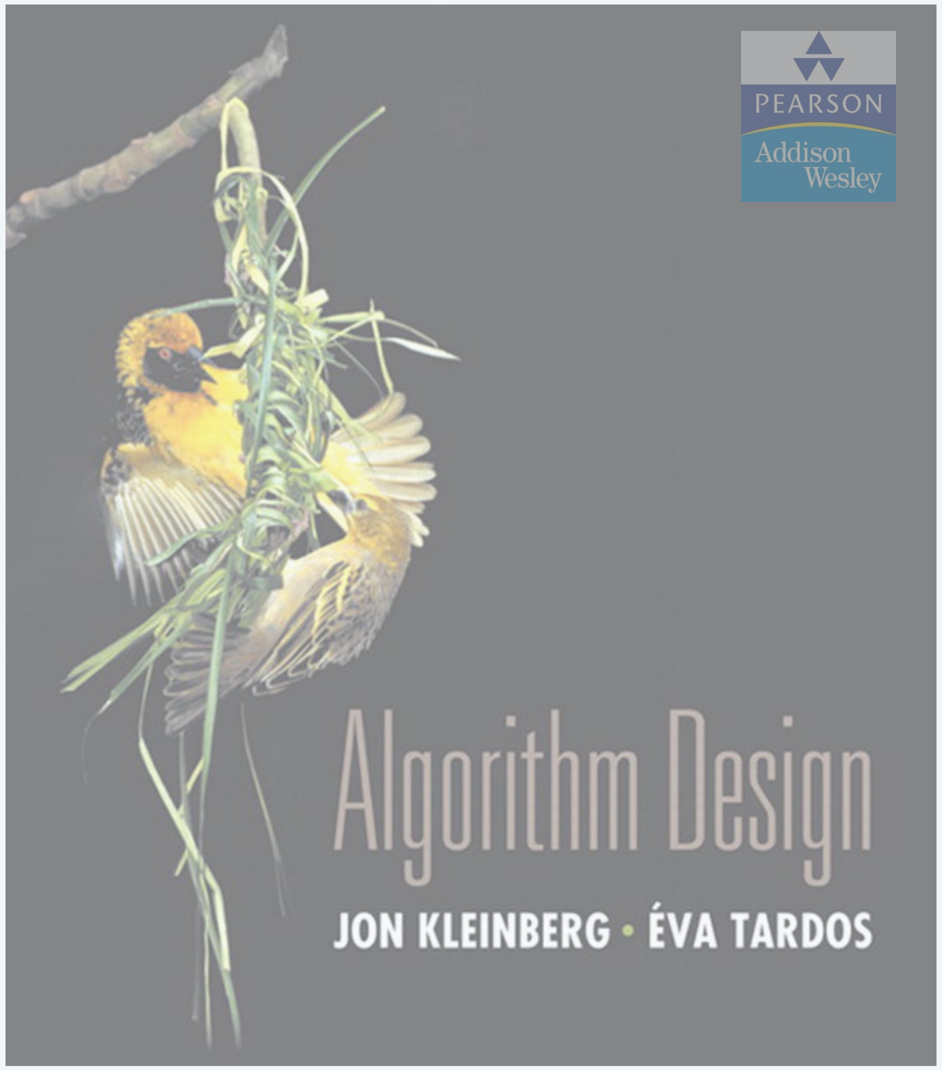


Remarks.

- Appel–Haken yields $O(n^4)$ algorithm to 4-color of a planar map.
- Best known: $O(n^2)$ to 4-color; $O(n)$ to 5-color.
- Determining whether 3 colors suffice is **NP**-complete.

Polynomial-time reductions



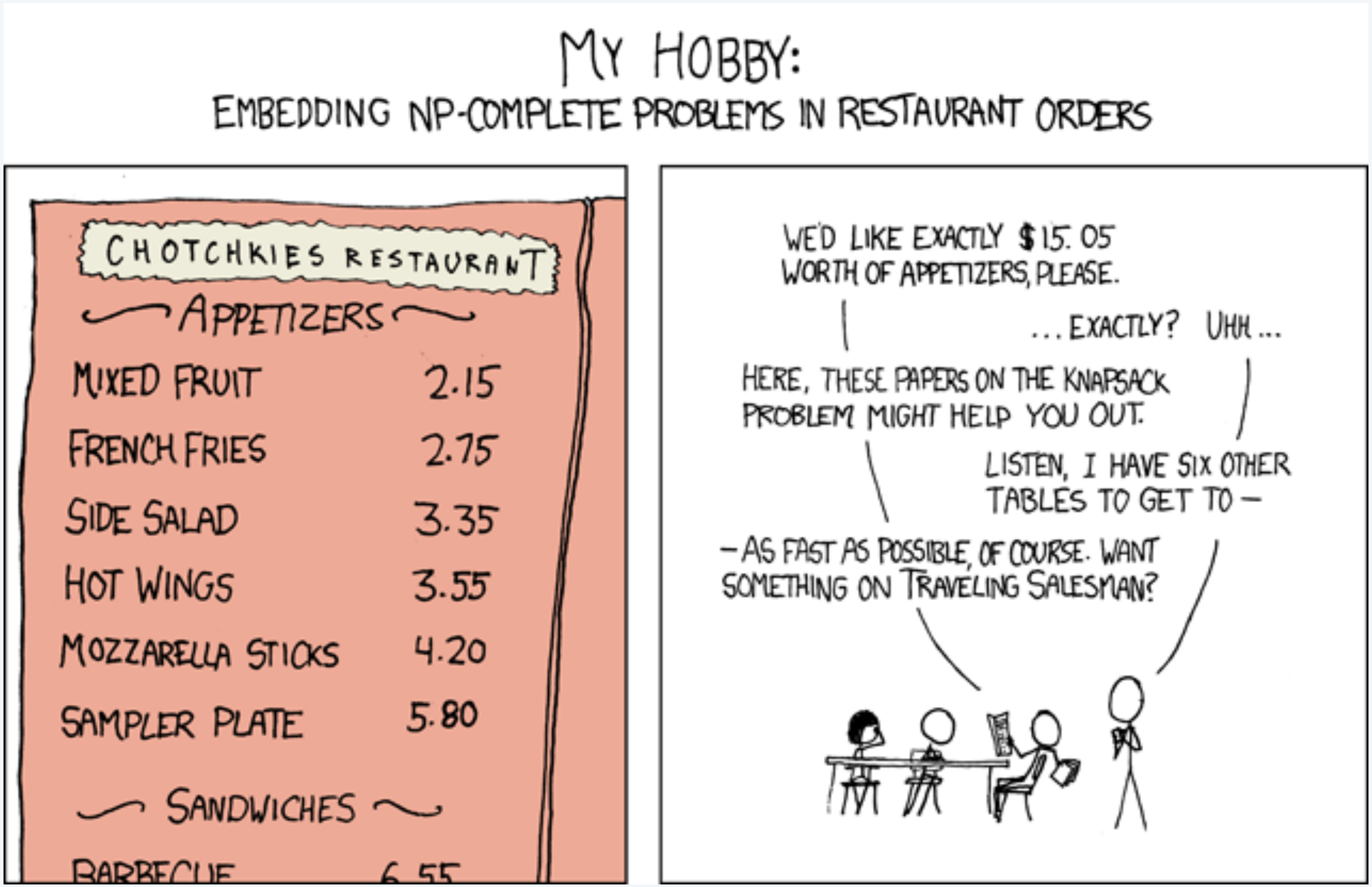


SECTION 8.8

8. INTRACTABILITY I

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

My hobby



NP-Complete by Randall Munro
http://xkcd.com/287

Subset sum

SUBSET-SUM. Given n natural numbers $w_1, \ldots, w_n$ 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-SAT $\le_P$ SUBSET-SUM.

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

3-satisfiability reduces to subset sum

Construction. Given 3-SAT instance Φ 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.

$$
\begin{aligned}
C_1 &= \neg x_1 \lor x_2 \lor x_3 \\
C_2 &= x_1 \lor \neg x_2 \lor x_3 \\
C_3 &= \neg x_1 \lor \neg x_2 \lor \neg x_3
\end{aligned}
$$

3-SAT instance

	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
dummies to get clause columns to sum to 4	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. Φ is satisfiable iff there exists a subset that sums to W.

Pf. $\Rightarrow$ Suppose 3-SAT instance Φ 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 Φ 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. ▪

$$
\begin{aligned}
C_1 &= \neg x_1 \vee x_2 \vee x_3 \\
C_2 &= x_1 \vee \neg x_2 \vee x_3 \\
C_3 &= \neg x_1 \vee \neg x_2 \vee \neg x_3
\end{aligned}
$$

3-SAT instance

	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
dummies to get clause columns to sum to 4	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. Φ 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. ▪

	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
dummies to get clause columns to sum to 4	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

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

3-SAT instance

Subset sum reduces to knapsack

Subset-Sum. Given n natural numbers $w_1, \ldots, w_n$ 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 Subset-Sum $\leq_P$ Knapsack.

Pf. Given instance $(w_1, \ldots, w_n, W)$ of Subset-Sum, create Knapsack instance:

Partition

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

PARTITION. Given natural numbers $v_1, \ldots, v_m$, can they be partitioned into two subsets that add up to the same value $\frac{1}{2} \Sigma_i \, v_i$?

Theorem. SUBSET-SUM $\le_P$ PARTITION.

Pf. Let $W, w_1, \ldots, w_n$ be an instance of SUBSET-SUM.

- Create instance of PARTITION with $m = n + 2$ elements.
  - $v_1 = w_1, v_2 = w_2, \ldots, \; v_n = w_n, \quad v_{n+1} = 2 \, \Sigma_i \, w_i \; - W, \quad v_{n+2} = \Sigma_i \, w_i + W$
- Lemma: there exists a subset that sums to W iff there exists a partition since elements v_{n+1} and v_{n+2} cannot be in the same partition. ▪

$v_{n+1} = 2 \, \Sigma_i \, w_i \; - W$	W	**subset A**
$v_{n+2} = \Sigma_i \, w_i + W$	$\Sigma_i \, w_i \; - W$	**subset B**

Scheduling with release times

SCHEDULE. Given a set of n jobs with processing time t_j, release time r_j, and deadline d_j, is it possible to schedule all jobs on a single machine such that job j is processed with a contiguous slot of t_j time units in the interval $[r_j, d_j]$?

Ex.

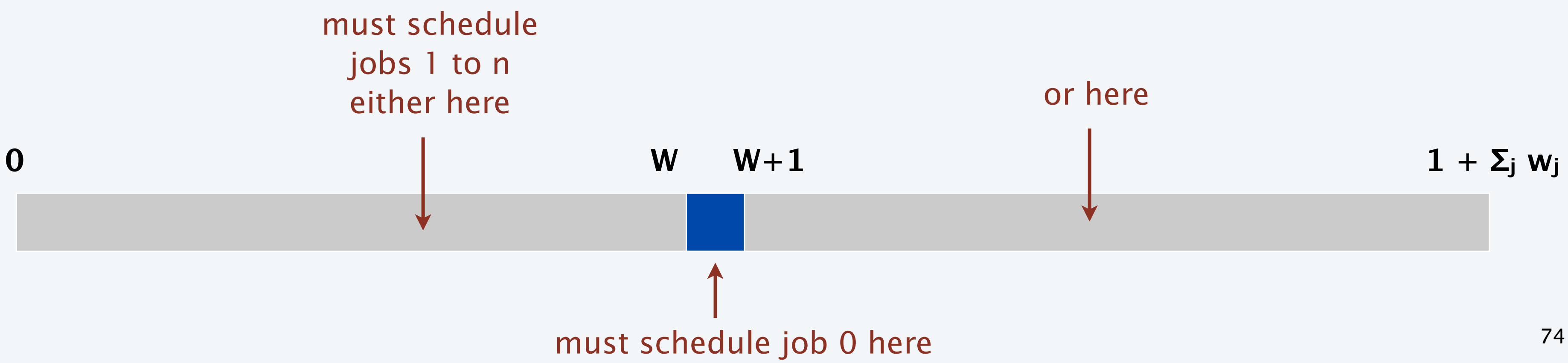
Scheduling with release times

Theorem. SUBSET-SUM $\leq_P$ SCHEDULE.

Pf. Given SUBSET-SUM instance $w_1, \ldots, w_n$ and target W, construct an instance of SCHEDULE that is feasible iff there exists a subset that sums to exactly W.

Construction.

- Create n jobs with processing time $t_j = w_j$, release time $r_j = 0$, and no deadline ($d_j = 1 + \Sigma_j w_j$).
- Create job 0 with $t_0 = 1$, release time $r_0 = W$, and deadline $d_0 = W + 1$.
- Lemma: subset that sums to W iff there exists a feasible schedule. ▪



Polynomial-time reductions