

Chapter 8

NP and Computational Intractability



Slides by Kevin Wayne.
Copyright © 2005 Pearson-Addison Wesley.
All rights reserved.

Algorithm Design Patterns and Anti-Patterns

Algorithm design patterns.

- Greed.
- Divide-and-conquer.
- Dynamic programming.
- Linear Programming
- Reductions.

Ex.

$O(n \log n)$ interval scheduling.

$O(n \log n)$ closest points.

$O(n^2)$ edit distance.

Algorithm design anti-patterns.

- NP-completeness.
- Undecidability.

$O(n^k)$ algorithm unlikely.

No algorithm possible.

8.1 Polynomial-Time Reductions

Classify Problems According to Computational Requirements

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

A working definition. [Cobham 1964, Edmonds 1965, Rabin 1966]

Those with **polynomial-time** algorithms.

Yes	Probably no
Shortest path	Longest path
Matching	3D-matching
Min cut	Max cut
2-SAT	3-SAT
Planar 4-color	Planar 3-color
Bipartite vertex cover	Vertex cover
Primality testing	Factoring

Polynomial-Time Reduction

Suppose we could solve Y in polynomial-time. What else could we solve in polynomial time?

Reduction. Problem X **polynomially 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 algorithm that solves problem Y .

Notation. $X \leq_p Y$.

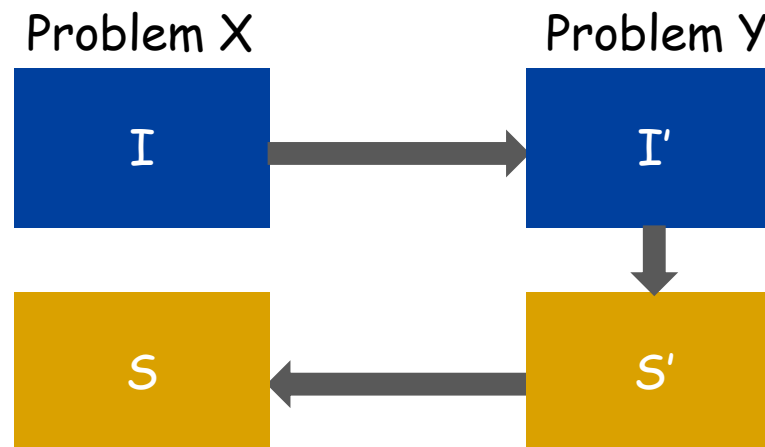
Remarks.

- We pay for time to write down instances sent to black box \Rightarrow instances of Y must be of polynomial size.

Polynomial-Time Reduction

Common way to establish polynomial time reduction from X to Y

1. Convert the instance I of problem X into an instance I' for problem Y in polynomial-time on the size of I
2. Obtain a solution S' for I' through a call to an algorithm that solves problem Y .
3. Convert the solution S' of instance I' into a solution S of instance I in polynomial-time on the size of I



Polynomial-Time Reduction

Purpose. Classify problems according to **relative** difficulty.

Design algorithms. If $X \leq_p Y$ and Y can be solved in polynomial-time, then X **can** also 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 $X \leq_p Y$ and $Y \leq_p X$, we use notation $X \equiv_p Y$.

↑
up to cost of reduction

Reduction By Simple Equivalence

Basic reduction strategies.

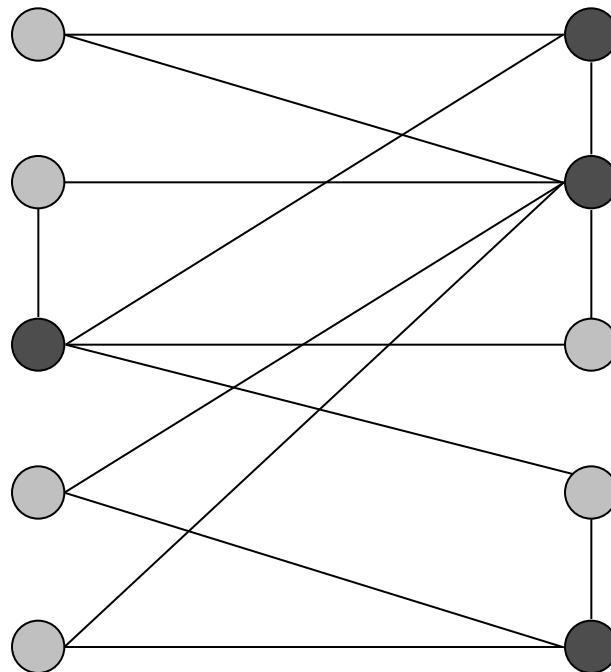
- Reduction by simple equivalence.
- Reduction from special case to general case.
- Reduction by encoding with gadgets.

Independent Set

INDEPENDENT SET: Given a graph $G = (V, E)$ and an integer k , is there a subset of vertices $S \subseteq V$ such that $|S| \geq k$, and for each edge at most one of its endpoints is in S ?

Ex. Is there an independent set of size ≥ 6 ? Yes.

Ex. Is there an independent set of size ≥ 7 ? No.



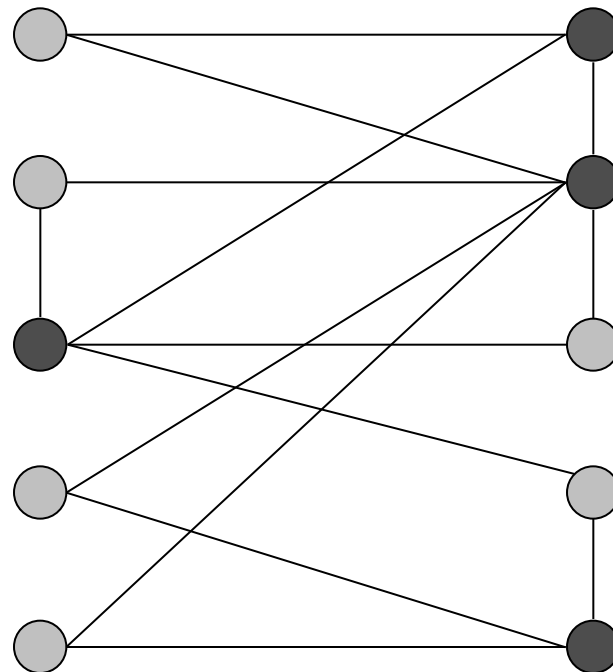
● independent set

Vertex Cover

VERTEX COVER: Given a graph $G = (V, E)$ and an integer k , is there a subset of vertices $S \subseteq V$ such that $|S| \leq k$, and for each edge, **at least one of its endpoints is in S** ?

Ex. Is there a vertex cover of size ≤ 4 ? Yes.

Ex. Is there a vertex cover of size ≤ 3 ? No.

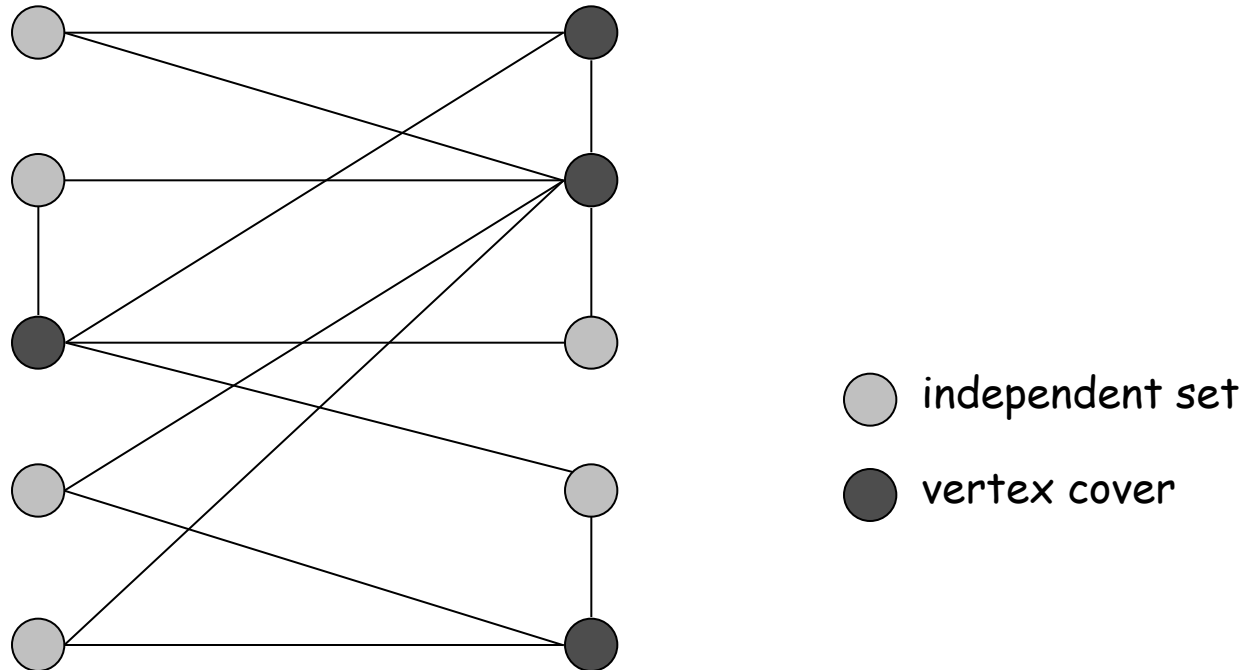


● vertex cover

Vertex Cover and Independent Set

Claim. VERTEX-COVER \equiv_p INDEPENDENT-SET.

Pf. We show S is an independent set iff $V - S$ is a vertex cover.



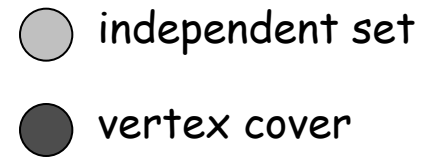
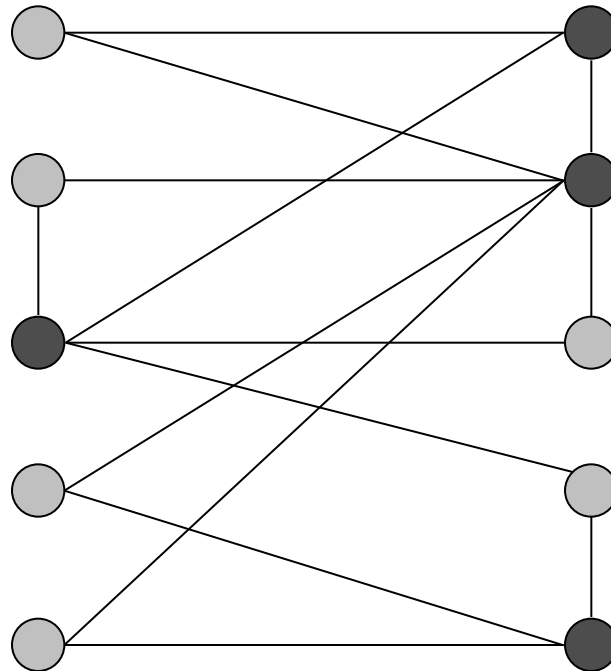
Vertex Cover and Independent Set

Claim. VERTEX-COVER \equiv_p INDEPENDENT-SET.

Pf. We show S is an independent set iff $V - S$ is a vertex cover.

\Rightarrow

- Let S be any independent set.
- Consider an arbitrary edge (u, v) .
- S independent $\Rightarrow u \notin S$ or $v \notin S \Rightarrow u \in V - S$ or $v \in V - S$.
- Thus, $V - S$ covers (u, v) .



Vertex Cover and Independent Set

Claim. VERTEX-COVER \equiv_p INDEPENDENT-SET.

Pf. We show S is an independent set iff $V - S$ is a vertex cover.

\Rightarrow

- Let S be any independent set.
- Consider an arbitrary edge (u, v) .
- S independent $\Rightarrow u \notin S$ or $v \notin S \Rightarrow u \in V - S$ or $v \in V - S$.
- Thus, $V - S$ covers (u, v) .

\Leftarrow

- Let $V - S$ be any vertex cover.
- Consider two nodes $u \in S$ and $v \in S$.
- Observe that $(u, v) \notin E$ since $V - S$ is a vertex cover.
- Thus, no two nodes in S are joined by an edge $\Rightarrow S$ independent set. ■

Independent Set to Vertex Cover

Solving INDEPENDENT SET using VERTEX COVER

1. Transform the instance (G, k) for Independent Set into the instance $(G, n-k)$ for Vertex Cover
2. Call VC algorithm to solve $(G, n-k)$
3. If the algorithm answer YES for VC then answer YES for Independent Set. Otherwise, answer NO.

Reduction from Special Case to General Case

Basic reduction strategies.

- Reduction by simple equivalence.
- Reduction from special case to general case.
- Reduction by encoding with gadgets.

Set Cover

SET COVER: Given a set U of elements, a collection S_1, S_2, \dots, S_m of subsets of U , and an integer k , does there exist a collection of $\leq k$ of these sets whose union is equal to U ?

Ex:

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

$$k = 2$$

$$S_1 = \{3, 7\}$$

$$S_4 = \{2, 4\}$$

$$S_2 = \{3, 4, 5, 6\}$$

$$S_5 = \{5\}$$

$$S_3 = \{1\}$$

$$S_6 = \{1, 2, 6, 7\}$$

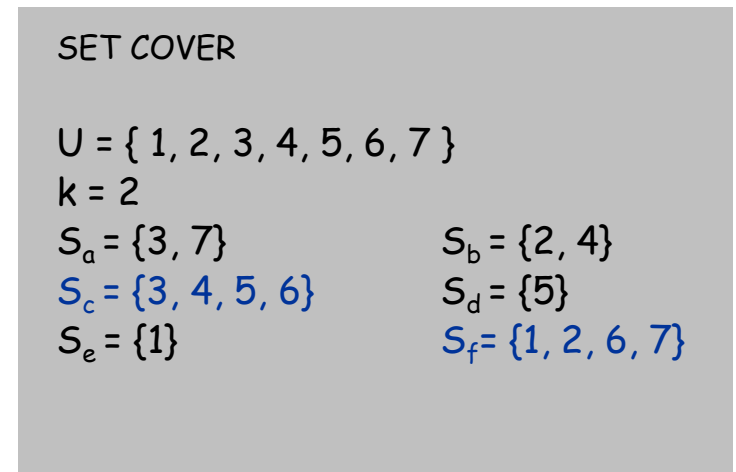
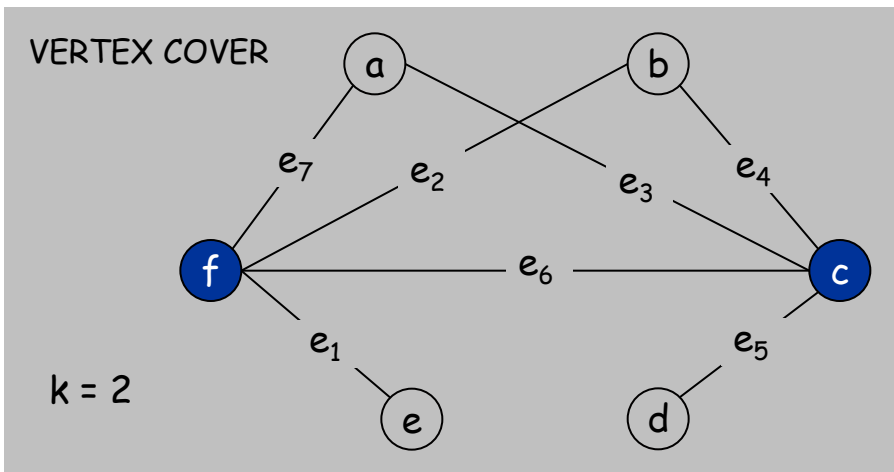
Vertex Cover Reduces to Set Cover

Claim. $\text{VERTEX-COVER} \leq_p \text{SET-COVER}$.

Pf. Given a VERTEX-COVER instance (G, k) , we construct a SET-COVER instance with one-to-one correspondence between solutions

Construction.

- Create SET-COVER instance:
 $k = k$, $U = E$, $S_v = \{e \in E : e \text{ incident to } v\}$
- Set-cover of size $\leq k$ iff vertex cover of size $\leq k$. ■



8.2 Reductions via "Gadgets"

Basic reduction strategies.

- Reduction by simple equivalence.
- Reduction from special case to general case.
- Reduction via "gadgets."

Satisfiability

3-CNF: A formula on boolean variables of the form "AND's of [OR's of three variables or their complements]"

3-SAT Problem: Given a 3-CNF, is there a setting of the variables that makes the formula TRUE? (i.e. is the formula *satisfiable*?)

Ex:
$$(\overline{x_1} \vee x_2 \vee x_3) \wedge (x_1 \vee \overline{x_2} \vee x_3) \wedge (x_2 \vee x_3 \vee x_4) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_3})$$

Yes: $x_1 = \text{true}, x_2 = \text{true}, x_3 = \text{false}.$

clause

3 Satisfiability Reduces to Independent Set

Claim. $3\text{-SAT} \leq_p \text{INDEPENDENT-SET}$.

Pf. Given an instance Φ of 3-SAT with k clauses, we construct an instance (G, k) of INDEPENDENT-SET that has an independent set of size k iff Φ is satisfiable.

$k = 3$

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

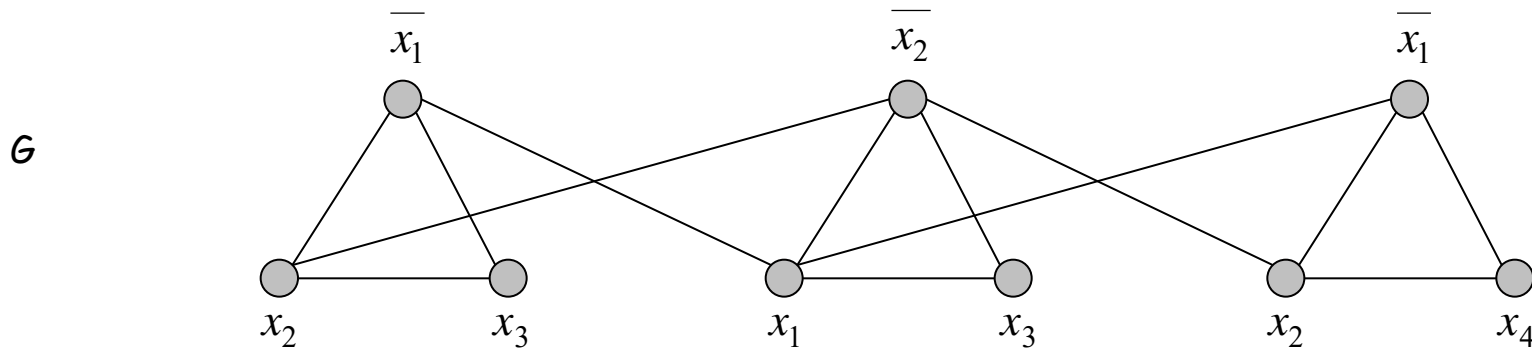
3 Satisfiability Reduces to Independent Set

Claim. $3\text{-SAT} \leq_p \text{INDEPENDENT-SET}$.

Pf. Given an instance Φ of 3-SAT with k clauses, we construct an instance (G, k) of INDEPENDENT-SET that has an independent set of size k iff Φ is satisfiable.

Construction.

- G contains 3 vertices for each clause, one for each literal.
- Connect 3 literals in a clause in a triangle.
- Connect literal to each of its negations.



$k = 3$

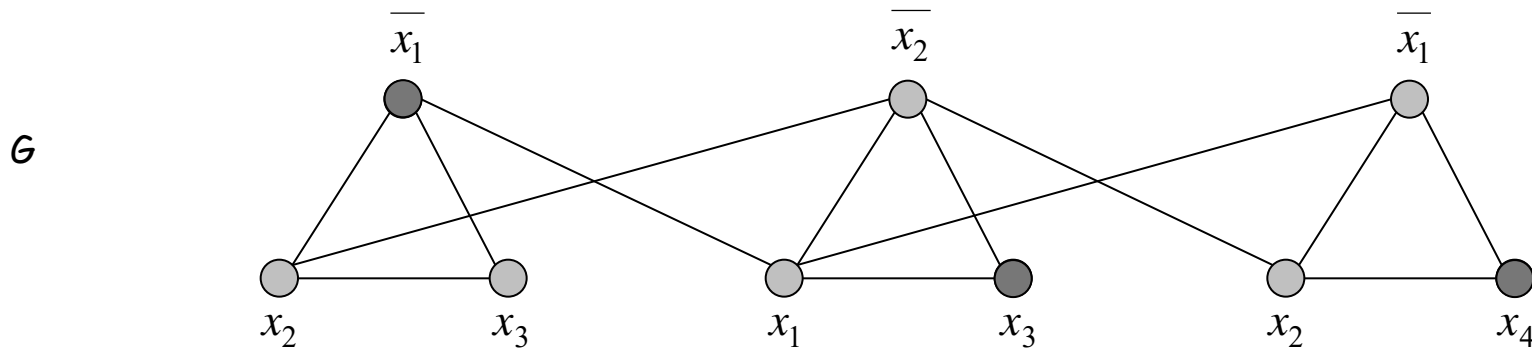
$$\Phi = (\overline{x_1} \vee x_2 \vee x_3) \wedge (x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee x_2 \vee x_4)$$

3 Satisfiability Reduces to Independent Set

Claim. G contains independent set of size $k = |\Phi|$ iff Φ is satisfiable.

Pf. \Rightarrow Let S be independent set of size k .

- S must contain exactly one vertex in each triangle.
- Set these literals to true. \leftarrow and any other variables in a consistent way
- Truth assignment is consistent and all clauses are satisfied.



$k = 3$

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

3 Satisfiability Reduces to Independent Set

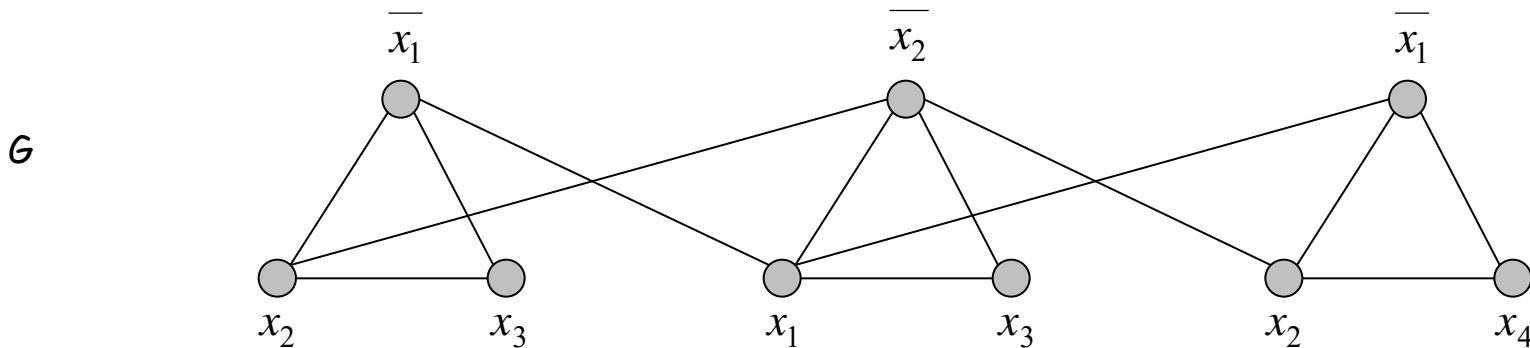
Claim. G contains independent set of size $k = |\Phi|$ iff Φ is satisfiable.

Pf. \Rightarrow Let S be independent set of size k .

- S must contain exactly one vertex in each triangle.
- Set these literals to true. \leftarrow and any other variables in a consistent way
- Truth assignment is consistent and all clauses are satisfied.

Pf \Leftarrow Suppose the formula has a satisfying assignment

- Given satisfying assignment, select one true literal from each triangle. This is an independent set of size k . ■



$k = 3$

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

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

Self-Reducibility

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

Search problem. **Find** vertex cover of minimum cardinality.

Clear: decision version \leq_p search version

Self-reducibility. Search problem \leq_p decision version.

- Applies to all (NP-complete) problems in this chapter.
- Justifies our focus on decision problems.

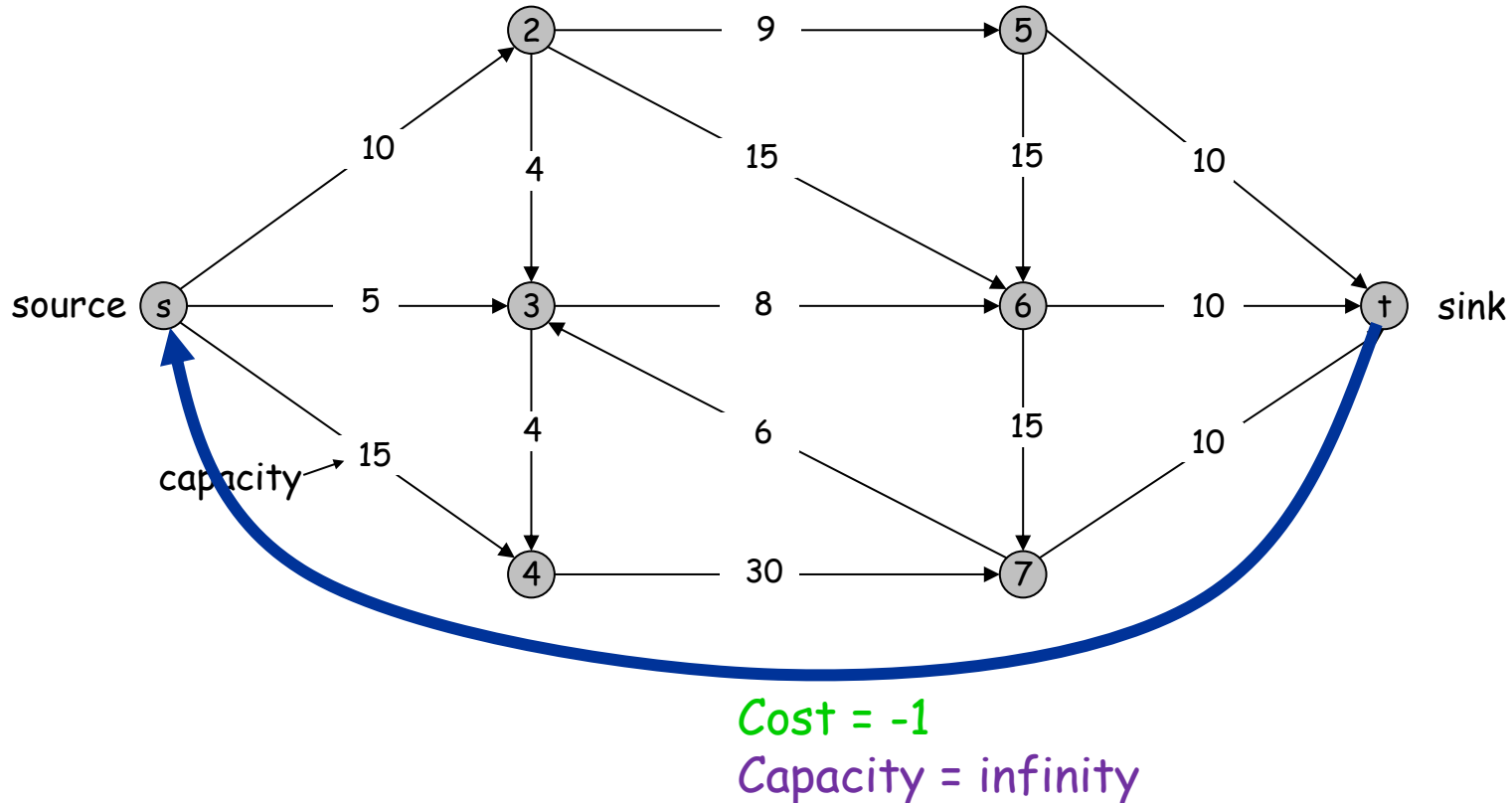
Ex: to find min cardinality vertex cover.

- (Binary) search for cardinality k^* of min vertex cover.
- Find a vertex v such that $G - \{v\}$ has a vertex cover of size $\leq k^* - 1$.
 - any vertex in any min vertex cover will have this property
- Include v in the vertex cover.
- Recursively find a min vertex cover in $G - \{v\}$.

↑
delete v and all incident edges

Exercise

Exercise: Reduce the Max-flow problem to the Min-cost flow problem



Add edge from sink to source with negative cost and infinite capacity (put cost 0 on all other edges), all demands/supply equal to 0

8.3 Definition of NP

Decision Problems

Decision problem.

- X is a set of strings.
- Instance: string s .
- Algorithm A solves problem X : $A(s) = \text{yes}$ iff $s \in X$.

Polynomial time. Algorithm A runs in poly-time if for every string s , $A(s)$ terminates in at most $p(|s|)$ "steps", where $p(\cdot)$ is some polynomial.

↑
length of s

PRIMES: $X = \{ 2, 3, 5, 7, 11, 13, 17, 23, 29, 31, 37, \dots \}$

Algorithm. [Agrawal-Kayal-Saxena, 2002] $p(|s|) = |s|^8$.

INDEPENDENT SET :

$X = \{ \text{family of graph with an independent set of size at least } k \}$

Algorithm. A polynomial algorithm is not known

Definition of P

P. Decision problems for which there is a poly-time algorithm.

The decision version of most problems we saw on the course are in P

Problem	Description	Algorithm	Yes	No
MULTIPLE	Is x a multiple of y?	Grade school division	51, 17	51, 16
RELPRIME	Are x and y relatively prime?	Euclid (300 BCE)	34, 39	34, 51
PRIMES	Is x prime?	AKS (2002)	53	51
EDIT-DISTANCE	Is the edit distance between x and y less than 5?	Dynamic programming	niether neither	acgggt ttttta
LSOLVE	Is there a vector x that satisfies $Ax = b$?	Gauss-Edmonds elimination	$\left[\begin{array}{ccc c} 0 & 1 & 1 & 4 \\ 2 & 4 & -2 & 2 \\ 0 & 3 & 15 & 36 \end{array} \right]$	$\left[\begin{array}{ccc c} 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{array} \right]$

NP

NP (informal). Problems that can be **certified** in poly-time

Certification algorithm intuition.

- Certifier views things from "managerial" viewpoint.
- Certifier doesn't determine whether $s \in X$ on its own; rather, it checks a proposed proof t that $s \in X$.

Def. 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 $C(s, t) = \text{yes}$.

↖
"certificate" or "witness"

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

↑
 $C(s, t)$ is a poly-time algorithm and
 $|t| \leq p(|s|)$ for some polynomial $p(\cdot)$.

Remark. NP stands for **nondeterministic** polynomial-time.

Certifiers and Certificates: 3-Satisfiability

SAT. Given a CNF formula Φ , is there a satisfying assignment?

Certificate.

Certifier.

Ex. $(\overline{x_1} \vee x_2 \vee x_3) \wedge (x_1 \vee \overline{x_2} \vee x_3) \wedge (x_1 \vee x_2 \vee x_4) \wedge (\overline{x_1} \vee \overline{x_3} \vee \overline{x_4})$

instance s

$x_1 = 1, x_2 = 1, x_3 = 0, x_4 = 1$

certificate t

Conclusion. SAT is in NP.

Certifiers and Certificates: 3-Satisfiability

SAT. Given a CNF formula Φ , is there a satisfying assignment?

Certificate. An assignment of truth values to the n boolean variables.

Certifier. Check that each clause in Φ has at least one true literal.

Ex.
$$(\overline{x_1} \vee x_2 \vee x_3) \wedge (x_1 \vee \overline{x_2} \vee x_3) \wedge (x_1 \vee x_2 \vee x_4) \wedge (\overline{x_1} \vee \overline{x_3} \vee \overline{x_4})$$

instance s

$$x_1 = 1, x_2 = 1, x_3 = 0, x_4 = 1$$

certificate t

Conclusion. SAT is in NP.

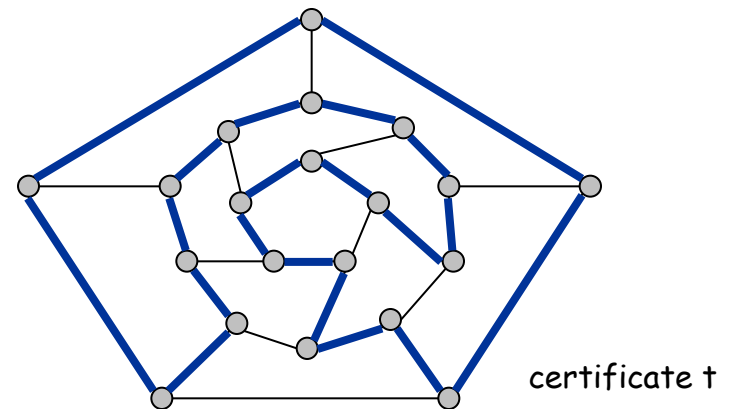
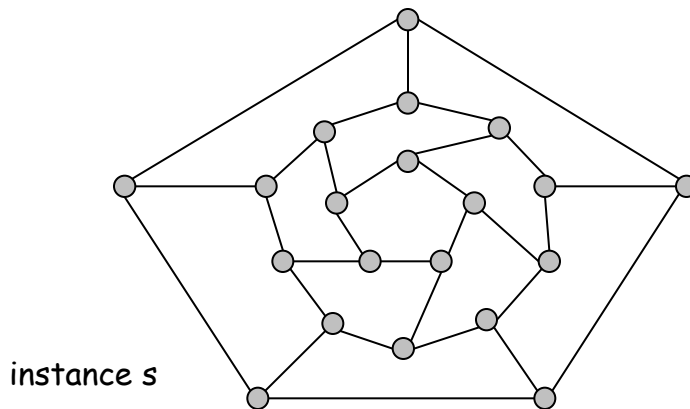
Certifiers and Certificates: Hamiltonian Cycle

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

Certificate.

Certifier.

Conclusion. HAM-CYCLE is in NP.



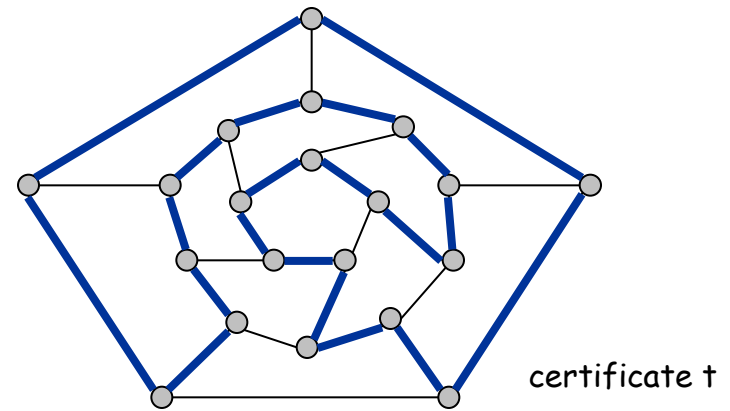
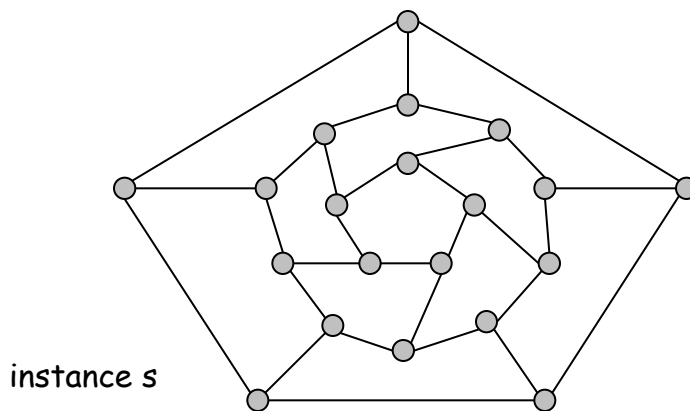
Certifiers and Certificates: Hamiltonian Cycle

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

Certificate. A permutation of the n nodes.

Certifier. Check that the permutation contains each node in V exactly once, and that there is an edge between each pair of adjacent nodes in the permutation.

Conclusion. HAM-CYCLE is in NP.



Certifiers and Certificates: COMPOSITE

COMPOSITE. Is x a composite number ?

Certificate. Two factors of x larger than 1.

Certifier. Verify if the two number are larger than 1 and if their product is x

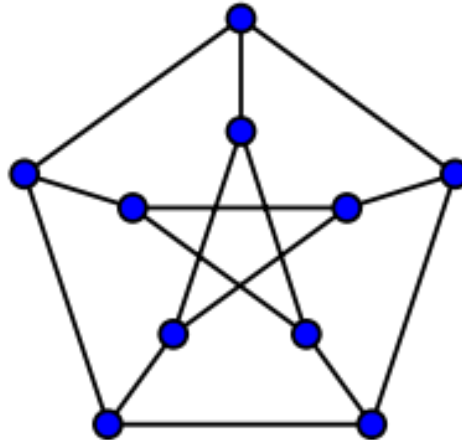
Conclusion. COMPOSITE is in NP.

Certifiers and Certificates: Non-Hamiltonian

Non-Hamiltonian. Is the graph G non-Hamiltonian?

Certificate.

It is believed that this problem is not in NP

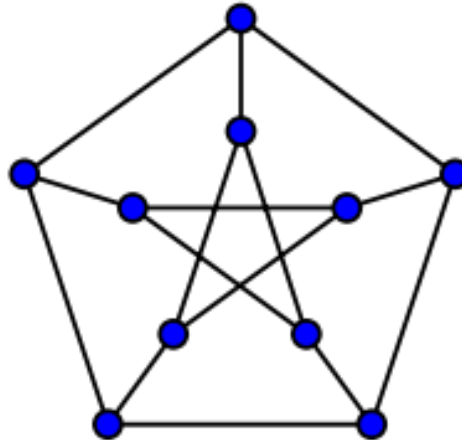


Certifiers and Certificates: Non-Hamiltonian

Non-Hamiltonian. Is the graph G non-Hamiltonian?

Certificate. ???

It is believed that this problem is not in NP



Certifiers and Certificates: PRIMES

PRIME. Is x a prime number ?

Certificate. empty

Certifier. AKS primality algorithm that runs in polynomial time

Conclusion. PRIMES is in NP.

The same reasoning allows us to conclude that $P \subseteq NP$
(that is, every problem with a polynomial-time algorithm
can be certified in polynomial time)

P, NP, EXP

P. Decision problems for which there is a **poly-time algorithm**.

EXP. Decision problems for which there is an **exponential-time algorithm**.

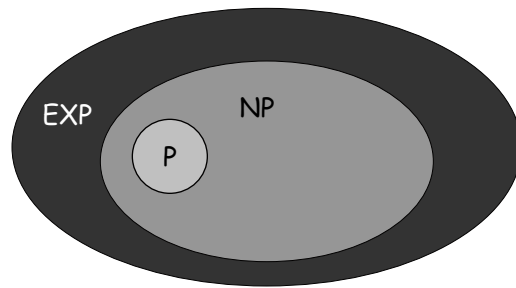
NP. Decision problems for which there is a **poly-time certifier**.

Claim. $P \subseteq NP \subseteq EXP$.

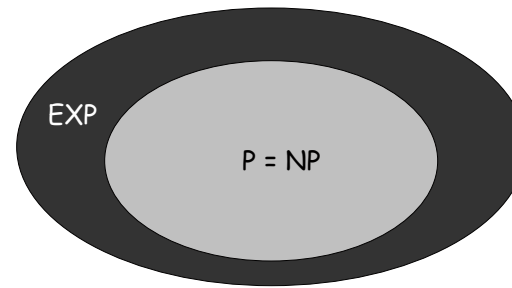
The Main Question: P Versus NP

Does $P = NP$? [Cook 1971, Edmonds, Levin, Yablonski, Gödel]
(is finding the solution as easy as checking the solution?)

Clay \$1 million prize.



If $P \neq NP$



If $P = NP$

would break RSA cryptography
(and potentially collapse economy)



If yes: Efficient algorithms for 3-COLOR, TSP, FACTOR, SAT, ...

If no: No efficient algorithms possible for 3-COLOR, TSP, SAT, ...

Consensus opinion on $P = NP$? Probably no.

Progress on P vs NP?

Some ways of proving $P \neq NP$ do not work:

- Relativization [Baker, Gill, Solovay '75]
- Natural Proofs [Rudich, Razborov '97]
- Algebrization [Aaronson, Wigderson '08]

Some ways of proving $P = NP$ do not work:

- Extended formulations [Yannakakis '91, Fiorini et al. '12, Rothvoss '14]

Maybe cannot be proved using standard math (i.e. independent of ZFC)

- If $P \neq NP$, some independence proofs would not work
[Ben-David, Halevi '92]

8.4 NP-Completeness

NP-Complete

“Hardest” NP problems

NP-complete. Problem Y is *NP-complete* if it is in NP and for **every** problem X in NP, $X \leq_p Y$.

Not clear they exist, could have harder and harder problems in NP

Theorem. Suppose Y is an NP-complete problem. Then Y is solvable in poly-time iff $P = NP$.

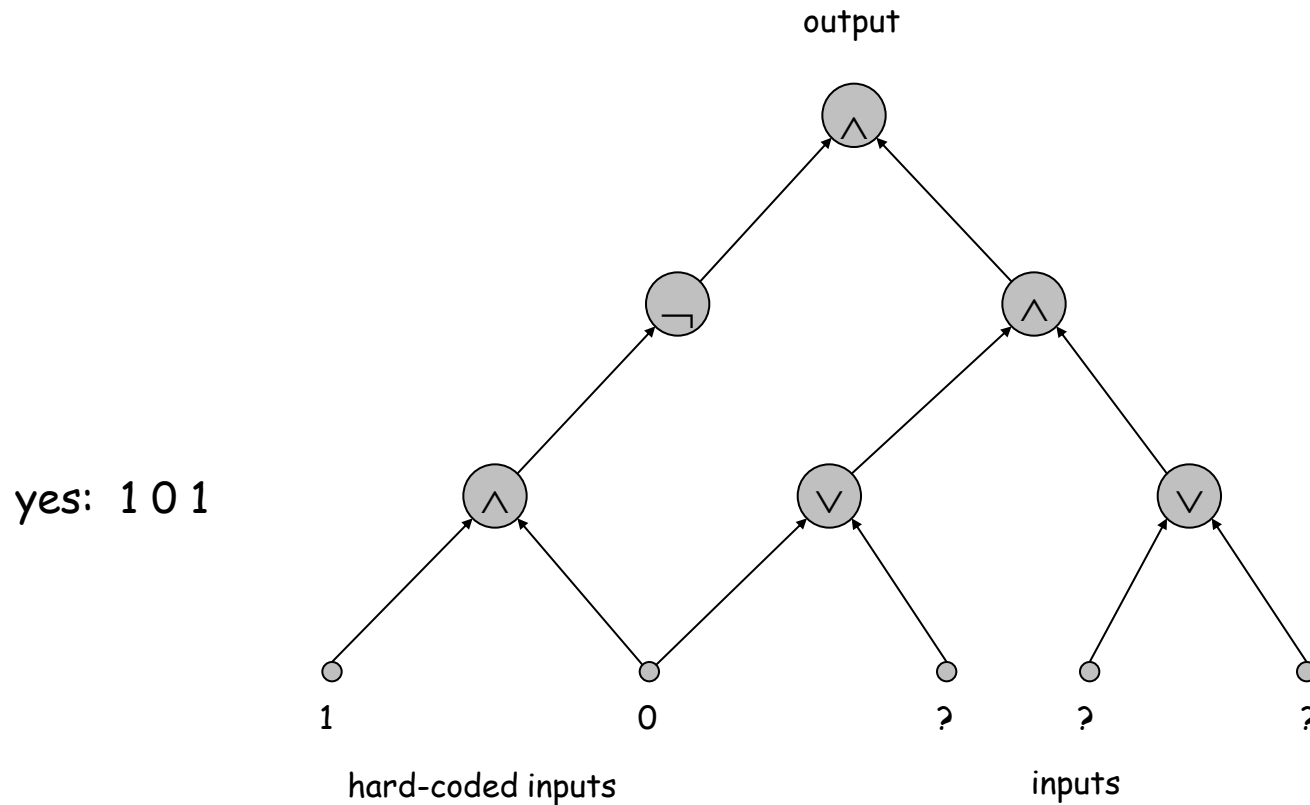
Pf. \Leftarrow If $P = NP$ then Y can be solved in poly-time since Y is in NP.

Pf. \Rightarrow Suppose Y can be solved in poly-time.

- Let X be any problem in NP. Since $X \leq_p Y$, we can solve X in poly-time. This implies $NP \subseteq P$.
- We already know $P \subseteq NP$. Thus $P = NP$. ■

Circuit Satisfiability

CIRCUIT-SAT. Given a combinational circuit built out of AND, OR, and NOT gates, is there a way to set the circuit inputs so that the output is 1?



Theorem. CIRCUIIT-SAT is NP-complete. [Cook 1971, Levin 1973]

The "First" NP-Complete Problem

Theorem. CIRCUIT-SAT is NP-complete. [Cook 1971, Levin 1973]

Pf. (sketch)

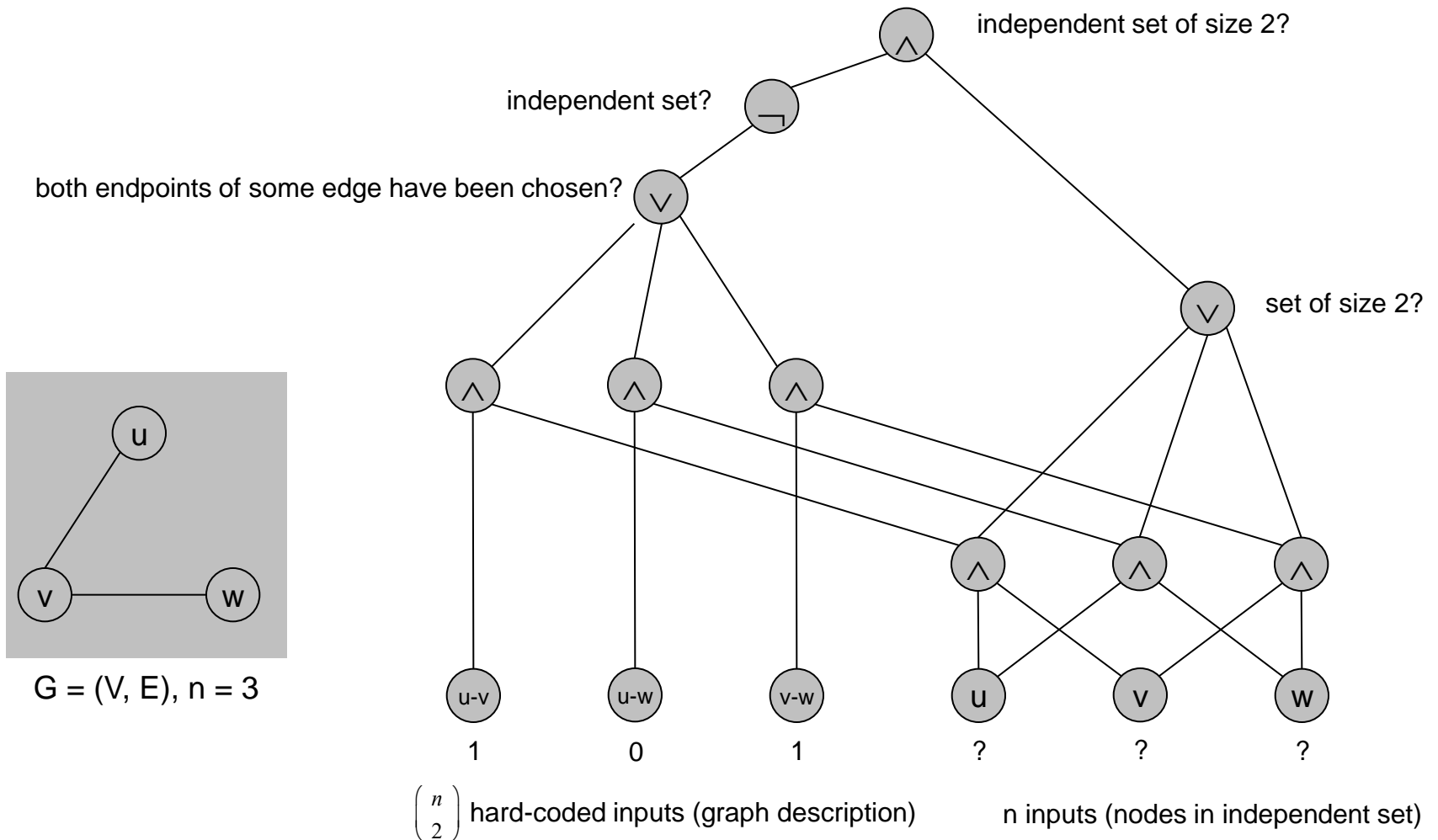
- Any algorithm that takes a fixed number n of bits as input and produces a yes/no answer can be represented by such a circuit. Moreover, if algorithm takes poly-time, then circuit is of poly-size.

sketchy part of proof; fixing the number of bits is important, and reflects basic distinction between algorithms and circuits

- Consider some problem X in NP. It has a poly-time certifier $C(s, t)$. To determine whether s is in X , need to know if there exists a certificate t of length $p(|s|)$ such that $C(s, t) = \text{yes}$
- View $C(s, t)$ as an algorithm on $|s| + p(|s|)$ bits (input s , certificate t) and convert it into a poly-size circuit K .
 - first $|s|$ bits are hard-coded with s
 - remaining $p(|s|)$ bits represent bits of t
- Circuit K is satisfiable iff there is certificate t for which $C(s, t) = \text{yes}$.

Example

Ex. Construction below creates a circuit K whose inputs can be set so that K outputs true iff graph G has an **independent set of size 2**.



Establishing NP-Completeness

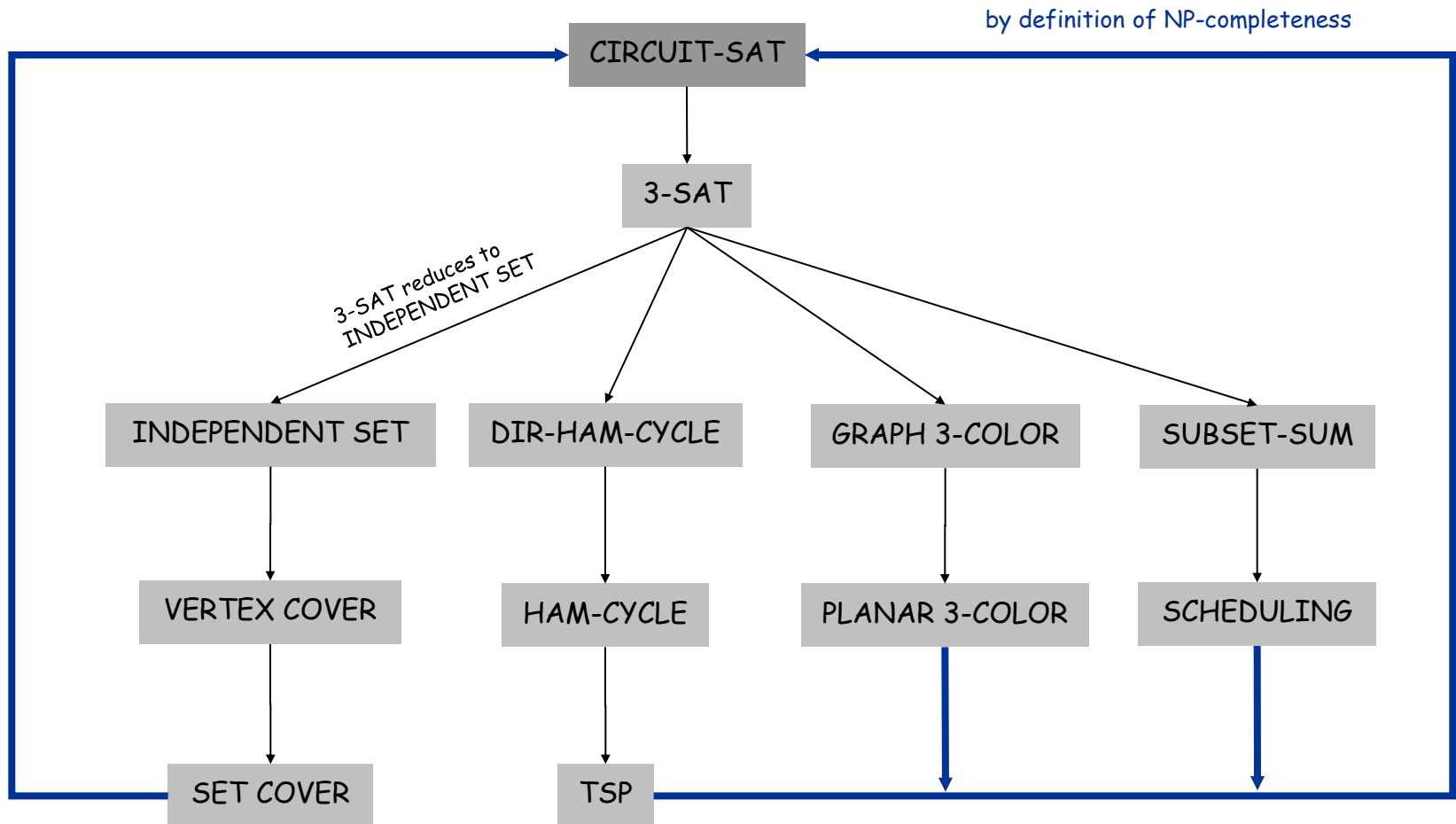
Remark. Once we establish first "natural" NP-complete problem, others fall like dominoes.

Recipe to establish NP-completeness of problem Y .

- Step 1. Show that Y is in NP.
- Step 2. Choose an NP-complete problem X .
- Step 3. Prove that $X \leq_p Y$. (so Y is as hard as the hardest NP problems)

NP-Completeness

Observation. All problems below are NP-complete and polynomially reduce to one another!



Extent and Impact of NP-Completeness

Extent of NP-completeness. [Papadimitriou 1995]

- Prime intellectual export of CS to other disciplines.
- 6,000 citations per year (title, abstract, keywords).
 - more than "compiler", "operating system", "database"
- Broad applicability and classification power.
- "Captures vast domains of computational, scientific, mathematical endeavors, and seems to roughly delimit what mathematicians and scientists had been aspiring to compute feasibly."

NP-completeness can guide scientific inquiry.

- If a problem is NP-complete, do not try to find a polynomial time algorithm
(would solve all NP problems, many of which very smart people worked on for a long time)

Coping With NP-Completeness

Q. Suppose I need to solve an NP-complete problem. What should I do?

Must sacrifice one of three desired features.

- Solve problem to optimality.
- Solve problem in polynomial time.
- Solve arbitrary instances of the problem.

Approaches to cope with NP-Completeness

- Approximation Algorithms:
- Exact Methods:
 - Branch and Bound, Integer Programming
- Heuristics:
 - Local Search, Simulated annealing, Genetic Algorithms
- Beyond worst-case analysis

.

Cursos Futuros

- ▣ Otimização Combinatória
 - Programação Linear
 - Algoritmos Aleatorizados
 - Algoritmos de Aproximação

- ▣ Programação Inteira

- ▣ Heurísticas/ Meta-Heurísticas