

# CS 381 – FALL 2019

Week 13.2, Wednesday, Nov 13

Homework 6 Due: November 14<sup>th</sup> 15<sup>th</sup> at 11:59PM (Gradescope)  
Homework 7 Released: November 15<sup>th</sup>

## 8.2 Reductions via "Gadgets"

---

Basic reduction strategies.

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

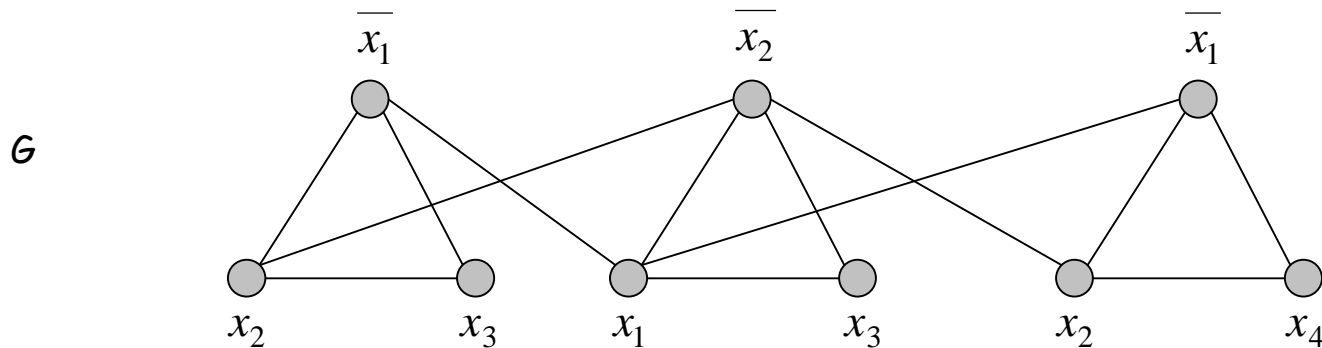
### 3 Satisfiability Reduces to Independent Set

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

**Pf.** Given an instance  $\Phi$  of 3-SAT, we construct an instance  $(G, k)$  of INDEPENDENT-SET that has an independent set of size  $k$  iff  $\Phi$  is satisfiable.

**Construction.**  $((G, k) = f(\Phi))$

- $G$  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.
- Set  $k = |\Phi|$  (number of clauses)



$k = 3$

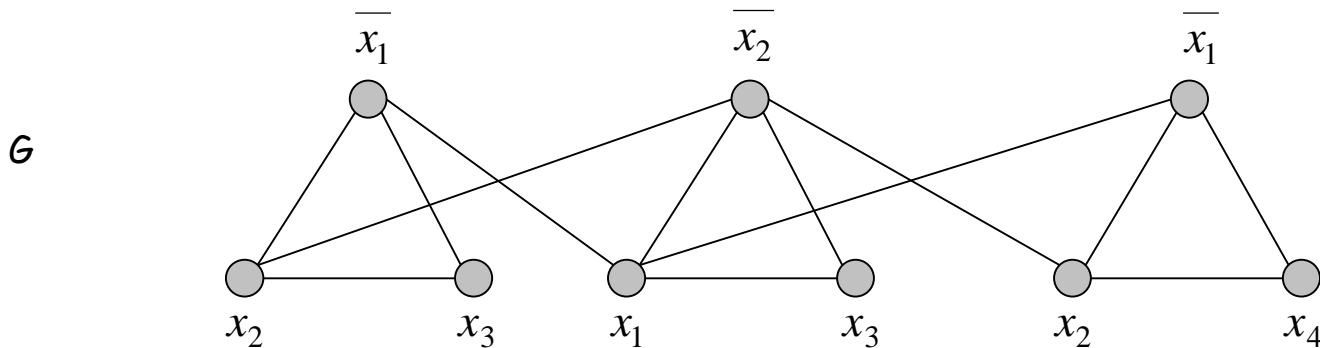
$$\Phi = (\overline{x_1} \vee x_2 \vee x_3) \wedge (x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee x_2 \vee x_4)$$

### 3 Satisfiability Reduces to Independent Set

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

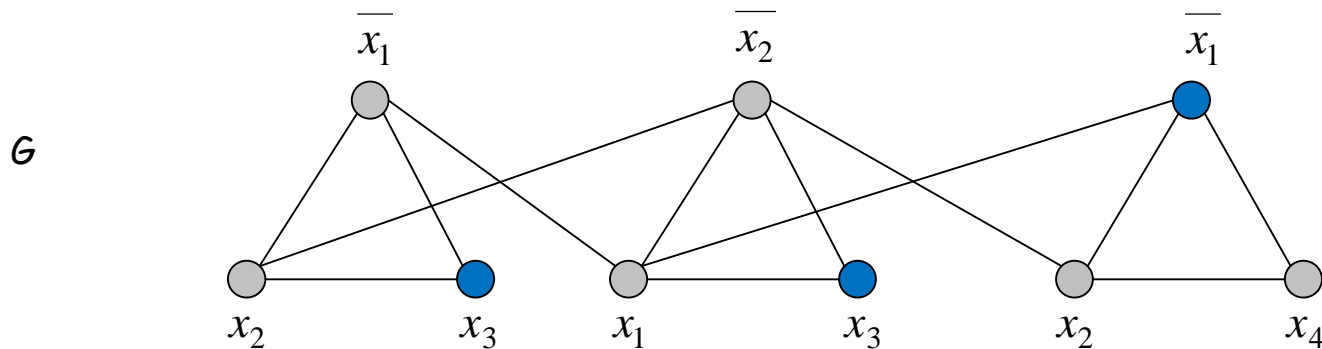
**Pf.**  $\Rightarrow$  Let  $S$  be independent set of size  $k$ .

- $S$  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.
- How do we know we don't assign  $x_i = \text{true}$  and  $\bar{x}_i = \text{false}$ ?

 Edges connecting nodes labeled  $x_i$  to nodes labeled  $\bar{x}_i$  ensure this

Independent Set  $S$  has size  $k=3$

$\rightarrow$  Set  $x_3 = \text{true}$ ,  $x_1 = \text{false}$  and  $x_2 = (\text{arbitrary})$



$k = 3$

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

### 3 Satisfiability Reduces to Independent Set

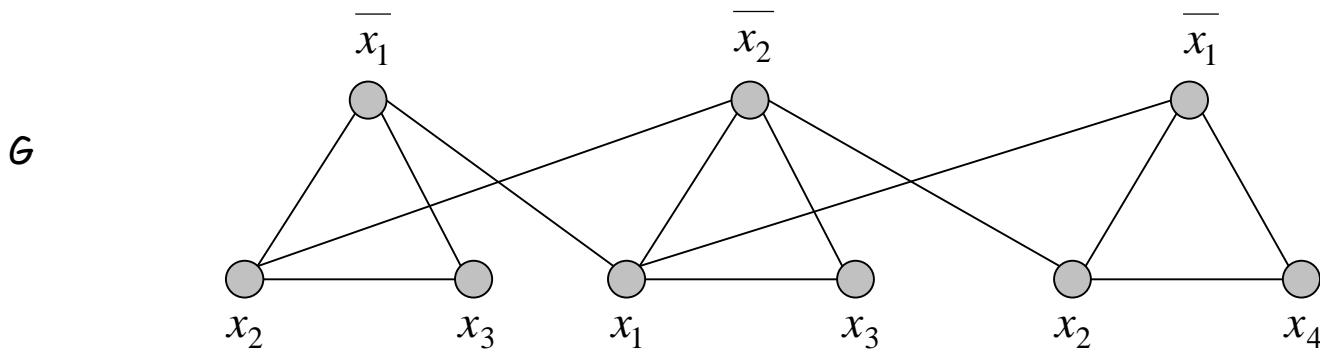
**Claim.**  $G$  contains independent set of size  $k = |\Phi|$  iff  $\Phi$  is satisfiable.

**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**  $\Leftarrow$  Given satisfying assignment, select one true literal from each triangle. This is an independent set of size  $k$ . (Why independent?)

- $x_i$  is true or false  $\rightarrow$  can never pick node labeled  $x_i$  and  $\bar{x}_i$  ■



$k = 3$

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

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

**Implication 1:** If we have a polynomial time algorithm for SET-COVER then we have a polynomial time algorithm for 3-SAT, INDEPENDENT-SET and VERTEX-COVER.

**Implication 2:** If there is no polynomial time algorithm for 3-SAT then there is no polynomial time algorithm for INDEPENDENT-SET, VERTEX-COVER or SET-COVER.

# Self-Reducibility

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

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

**Self-reducibility.** Search problem  $\leq_p$  decision version.

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

**Ex:** to find min cardinality vertex cover.

- (Binary) search for cardinality  $k^*$  of min vertex cover.
  - Oracle Query 1: Does  $G$  have a vertex cover of size  $k=n/2$ ? etc...
- 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





## 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]  $p(|s|) = |s|^8$ .

# Definition of P

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

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

## 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: Composite

**COMPOSITES.** Given an integer  $s$ , is  $s$  composite?

**Certificate.** A nontrivial factor  $t$  of  $s$ . Note that such a certificate exists iff  $s$  is composite. Moreover  $|t| \leq |s|$ .

**Certifier.**

```
boolean C(s, t) {  
    if (t ≤ 1 or t ≥ s)  
        return false  
    else if (s is a multiple of t)  
        return true  
    else  
        return false  
}
```

**Instance.**  $s = 437,669$ .

**Certificate.**  $t = 541$  or  $809$ .  $\longleftarrow 437,669 = 541 \times 809$

**Conclusion.** COMPOSITES is in NP.

## Certifiers and Certificates: 3-Satisfiability

**SAT.** Given a CNF formula  $\Phi$ , is there a satisfying assignment?

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

**Certifier.** Check that each clause in  $\Phi$  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.

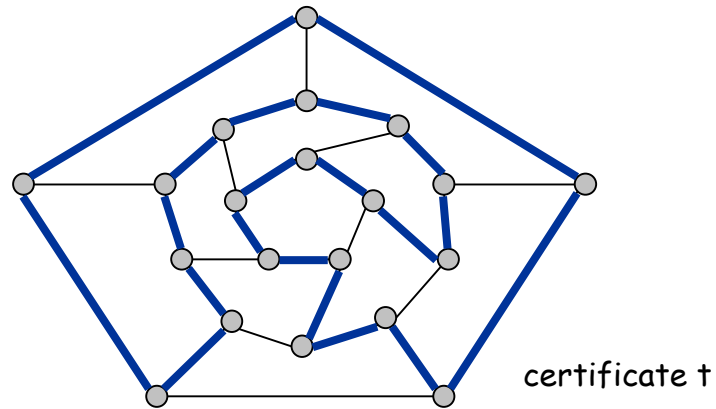
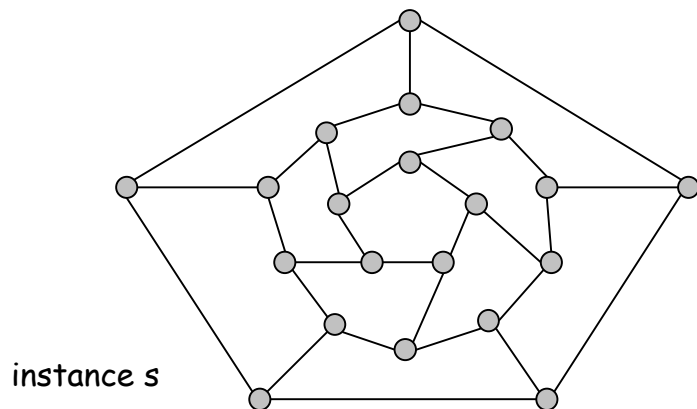
# Certifiers and Certificates: Hamiltonian Cycle

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

**Certificate.** 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.



# 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.**  $P \subseteq NP$ .

**Pf.** Consider any problem  $X$  in  $P$ .

- By definition, there exists a poly-time algorithm  $A(s)$  that solves  $X$ .
- Certificate:  $t = \varepsilon$ , certifier  $C(s, t) = A(s)$ . ▪

**Claim.**  $NP \subseteq EXP$ .

**Pf.** Consider any problem  $X$  in  $NP$ .

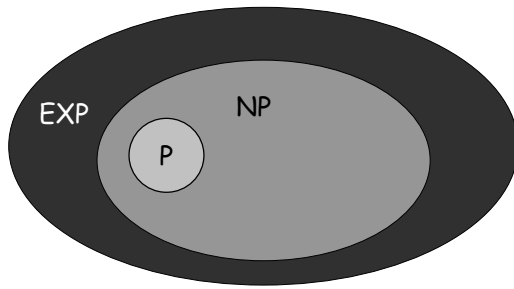
- By definition, there exists a poly-time certifier  $C(s, t)$  for  $X$ .
- To solve input  $s$ , run  $C(s, t)$  on all strings  $t$  with  $|t| \leq p(|s|)$ .
- Return  $_{yes}$ , if  $C(s, t)$  returns  $_{yes}$  for any of these. ▪



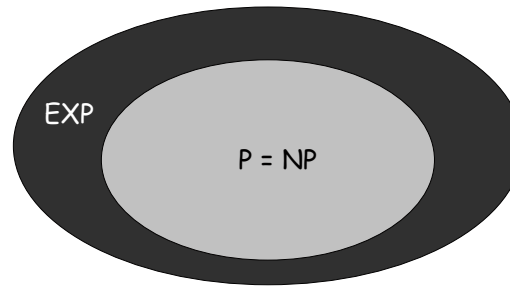
# The Main Question: P Versus NP

Does  $P = NP$ ? [Cook 1971, Edmonds, Levin, Yablonski, Gödel]

- Is the decision problem as easy as the certification problem?
- 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 not.



$P = NP ?$



## 8.4 NP-Completeness

---

# Polynomial Transformation

**Def.** Problem  $X$  **polynomial reduces** (Cook) 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  $Y$ .

**Def.** Problem  $X$  **polynomial transforms** (Karp) to problem  $Y$  if given any input  $x$  to  $X$ , we can construct an input  $y=f(x)$  such that  $x$  is a  $yes$  instance of  $X$  iff  $y$  is a  $yes$  instance of  $Y$ .

↑  
Reduction  $f(x)$  must be computable in polynomial time  $\rightarrow |y|$  has size polynomial in  $|x|$ ,

**Note.** Polynomial transformation is polynomial reduction with just one call to oracle for  $Y$ , exactly at the end of the algorithm for  $X$ . Almost all previous reductions were of this form.

**Open question.** Are these two concepts the same with respect to NP?

↑  
we abuse notation  $\leq_p$  and blur distinction

# NP-Complete

**NP-complete.** A problem  $Y$  in NP with the property that for every problem  $X$  in NP,  $X \leq_p Y$ .

**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.

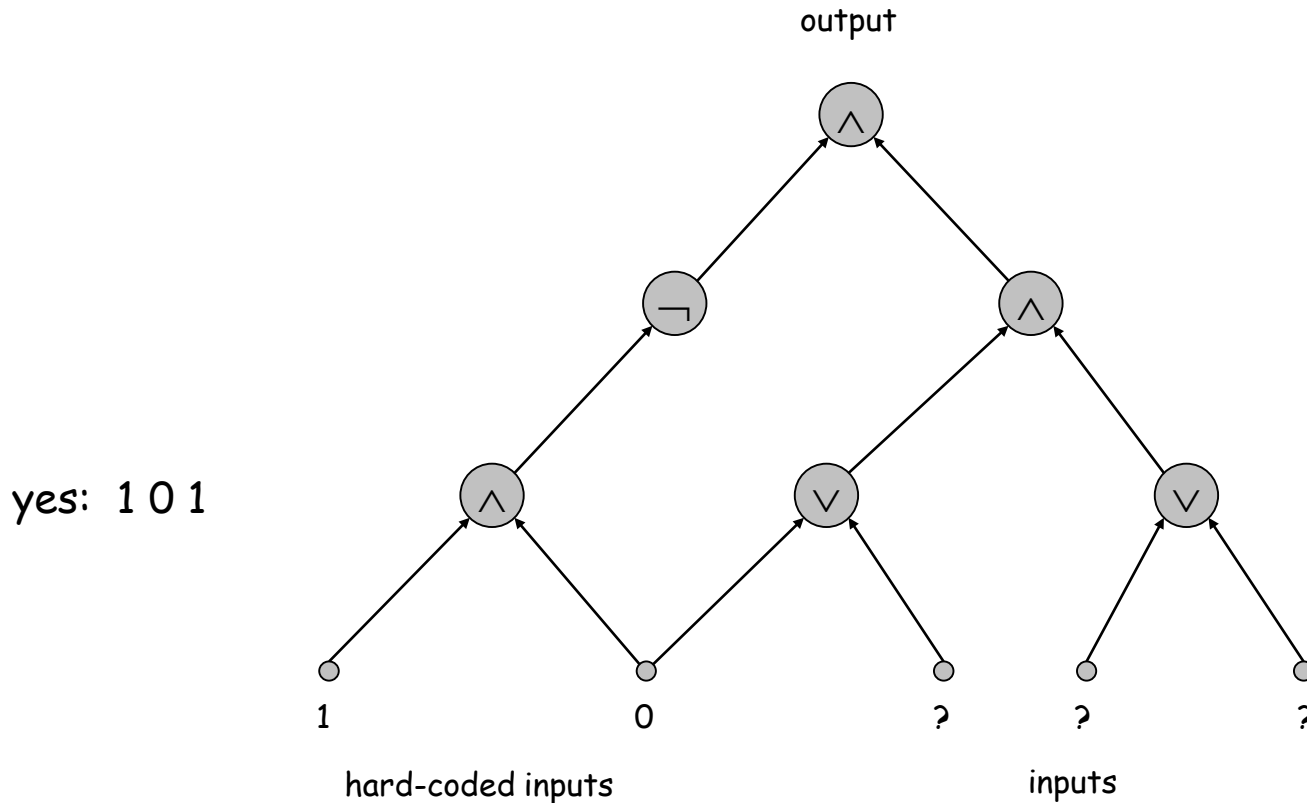
**Pf.**  $\Rightarrow$  Suppose  $Y$  can be solved in poly-time.

- 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  $P = NP$ . ▪

**Fundamental question.** Do there exist "natural" NP-complete problems?

# 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?



# The "First" NP-Complete Problem

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

**Pf.** (sketch)

- Any algorithm that takes a fixed number of bits  $n$  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.

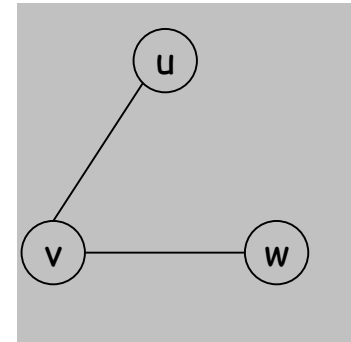
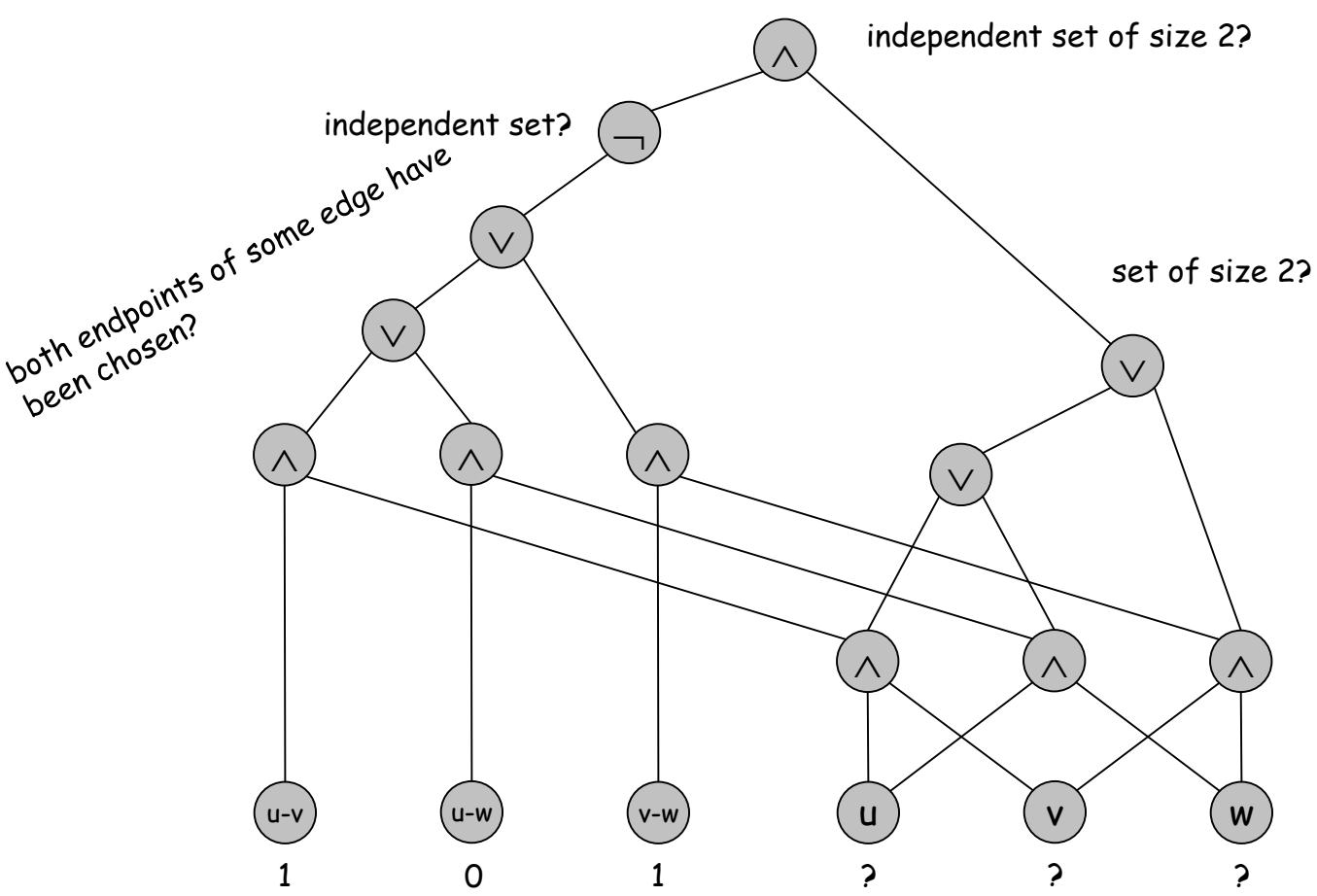
sketchy part of proof; fixing the number of bits is important, and reflects basic distinction between algorithms and circuits

- 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 exists a length  $p(|s|)$  bit input string  $t$  s.t.  $C(s, t) = \text{yes}$ .



# 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.



$G = (V, E), n = 3$

$\binom{n}{2}$  hard-coded inputs (graph description)

n inputs (nodes in independent set)

# 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 a known NP-complete problem  $X$ .
- Step 3. Prove that  $X \leq_p Y$ .

**Justification.** If  $X$  is an NP-complete problem, and  $Y$  is a problem in NP with the property that  $X \leq_p Y$  then  $Y$  is NP-complete.

**Pf.** Let  $W$  be any problem in NP. Then  $W \leq_p X \leq_p Y$ .

- By transitivity,  $W \leq_p Y$ .
- Hence  $Y$  is NP-complete. ▪

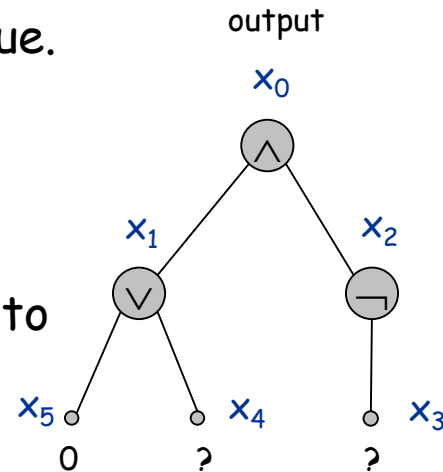
$\uparrow$                        $\uparrow$   
by definition of      by assumption  
NP-complete

# 3-SAT is NP-Complete

**Theorem.** 3-SAT is NP-complete.

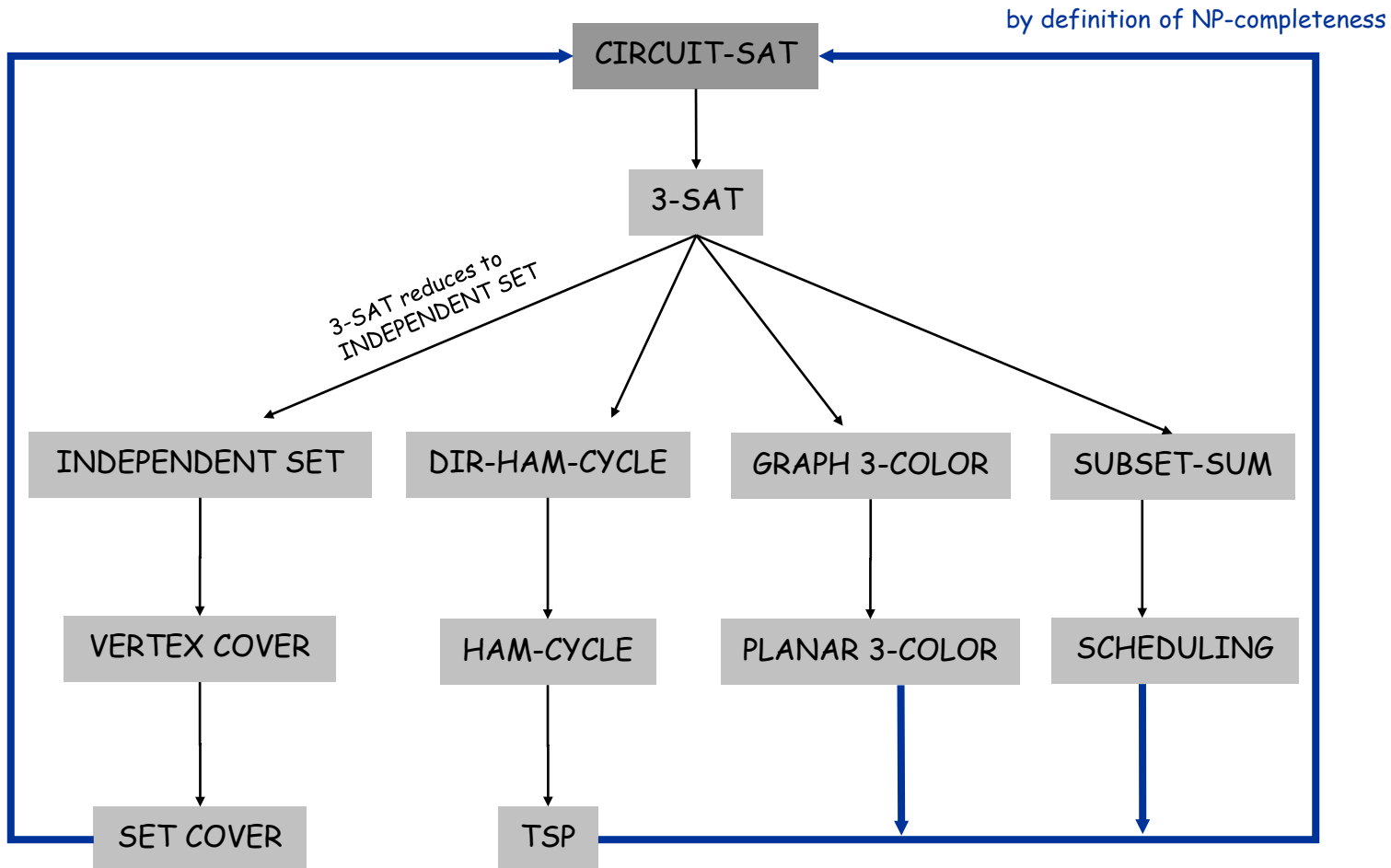
**Pf.** Suffices to show that  $CIRCUIT-SAT \leq_p 3-SAT$  since 3-SAT is in NP.

- Let  $K$  be any circuit.
- Create a 3-SAT variable  $x_i$  for each circuit element  $i$ .
- Make circuit compute correct values at each node:
  - $x_2 = \neg x_3 \Rightarrow$  add 2 clauses:  $x_2 \vee x_3, \overline{x_2} \vee \overline{x_3}$
  - $x_1 = x_4 \vee x_5 \Rightarrow$  add 3 clauses:  $x_1 \vee \overline{x_4}, x_1 \vee \overline{x_5}, \overline{x_1} \vee x_4 \vee x_5$
  - $x_0 = x_1 \wedge x_2 \Rightarrow$  add 3 clauses:  $\overline{x_0} \vee x_1, \overline{x_0} \vee x_2, x_0 \vee \overline{x_1} \vee \overline{x_2}$
- Hard-coded input values and output value.
  - $x_5 = 0 \Rightarrow$  add 1 clause:  $\overline{x_5}$
  - $x_0 = 1 \Rightarrow$  add 1 clause:  $x_0$
- Final step: turn clauses of length  $< 3$  into clauses of length exactly 3. ▪



# NP-Completeness

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



## Some NP-Complete Problems

Six basic genres of NP-complete problems and paradigmatic examples.

- Packing problems: SET-PACKING, INDEPENDENT SET.
- Covering problems: SET-COVER, VERTEX-COVER.
- Constraint satisfaction problems: SAT, 3-SAT.
- Sequencing problems: HAMILTONIAN-CYCLE, TSP.
- Partitioning problems: 3D-MATCHING 3-COLOR.
- Numerical problems: SUBSET-SUM, KNAPSACK.

**Practice.** Most NP problems are either known to be in P or NP-complete.

**Notable exceptions.** Factoring, graph isomorphism, Nash equilibrium.

# 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.

- 1926: Ising introduces simple model for phase transitions.
- 1944: Onsager solves 2D case in tour de force.
- 19xx: Feynman and other top minds seek 3D solution.
- 2000: Istrail proves 3D problem NP-complete.

# More Hard Computational Problems

**Aerospace engineering:** optimal mesh partitioning for finite elements.

**Biology:** protein folding.

**Chemical engineering:** heat exchanger network synthesis.

**Civil engineering:** equilibrium of urban traffic flow.

**Economics:** computation of arbitrage in financial markets with friction.

**Electrical engineering:** VLSI layout.

**Environmental engineering:** optimal placement of contaminant sensors.

**Financial engineering:** find minimum risk portfolio of given return.

**Game theory:** find Nash equilibrium that maximizes social welfare.

**Genomics:** phylogeny reconstruction.

**Mechanical engineering:** structure of turbulence in sheared flows.

**Medicine:** reconstructing 3-D shape from biplane angiogram.

**Operations research:** optimal resource allocation.

**Physics:** partition function of 3-D Ising model in statistical mechanics.

**Politics:** Shapley-Shubik voting power.

**Pop culture:** Minesweeper consistency.

**Statistics:** optimal experimental design.