Algorithm Design and Analysis

CSE 565

LECTURE 27

Computational Intractability

- Self-reducibility
- Classes P, NP, EXP
- NP-completeness

Sofya Raskhodnikova

Review

- Basic reduction strategies.
  - Simple equivalence: INDEPENDENT-SET $\equiv_{\text{P}}$ VERTEX-COVER.
  - Special case to general case: VERTEX-COVER $\leq_{\text{P}}$ SET-COVER.
  - Encoding with gadgets: 3-SAT $\leq_{\text{P}}$ INDEPENDENT-SET.

- **Transitivity**. If X $\leq_p$ Y and Y $\leq_p$ Z, then X $\leq_p$ Z.
- Proof idea. Compose the two algorithms.
- Ex: 3-SAT $\leq_p$ INDEPENDENT-SET $\leq_p$ VERTEX-COVER $\leq_p$ 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 Chapter 8 of KT.
  - 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

S. Raskhodnikova; based on slides by E. Demaine, C. Leiserson, A. Smith, K. Wayne

Definitions of P and NP

Decision Problems

Decision problem.

- X is a set of strings.
- Instance: string s.
- Algorithm A solves problem X: **A(s) = `yes`** iff **s ∈ 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(·) is some polynomial.

length of s

PRIMES: X = { 2, 3, 5, 7, 11, 13, 17, 23, 29, 31, 37, …. }

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

Definition of P

P. The class of decision problems for which there is a poly-time algorithm.

Examples

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	$\begin{bmatrix} 0 & 1 & 1 \\ 2 & 4 & -2 \\ 0 & 3 & 15 \end{bmatrix}, \begin{bmatrix} 4 \\ 2 \\ 36 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$

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 $C(s, t)$ = 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 $|t| \le |s|$.

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. ← 437,669 = 541 × 809

Conclusion. COMPOSITES 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.

$$\left(\overline{x_1} \vee x_2 \vee x_3\right) \wedge \left(x_1 \vee \overline{x_2} \vee x_3\right) \wedge \left(x_1 \vee x_2 \vee x_4\right) \wedge \left(\overline{x_1} \vee \overline{x_3} \vee \overline{x_4}\right)$$

instance s

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

certificate t

Conclusion. SAT is in NP.

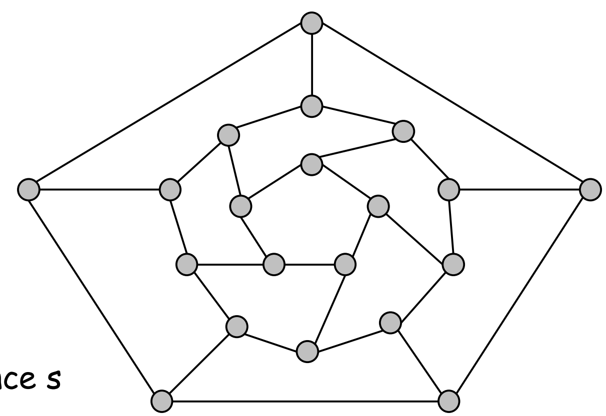
Certifiers and Certificates: Hamiltonian Cycle

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

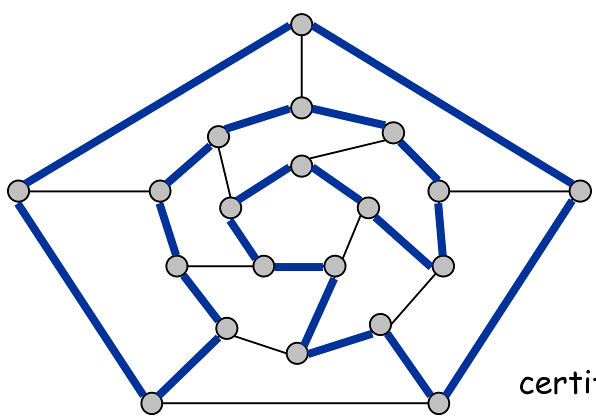
Certificate. A permutation of the n nodes.

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.



instance s



certificate t

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.

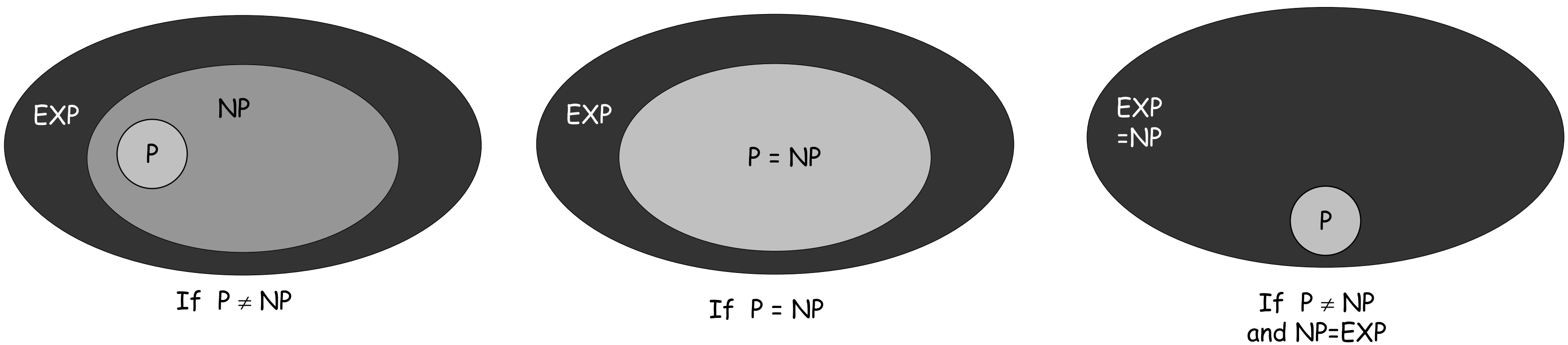
- By definition, there exists a poly-time algorithm $A(s)$ that solves X.
- Certificate: $t = \varepsilon$, certifier $C(s, t) = A(s)$. ▪

Claim. NP $\subseteq$ EXP.
Pf. Consider any problem X in NP.

- By definition, there exists a poly-time certifier $C(s, t)$ for X that runs in time $p(|s|)$.
- 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 Big Question: P vs. NP

- Does P = NP? [Cook 1971, Edmonds, Levin, Yablonski, Gödel]
  - Is the decision problem as easy as the verification problem?
  - Clay $1 million prize.



- If yes: Efficient algorithms for HamPath, SAT, TSP, factoring
  - Cryptography is impossible*
  - Creativity is automatable
- If no: No efficient algorithms possible for these problems.
- Consensus opinion on P = NP? Probably no.

NP-completeness

NP-Complete

NP-complete. A problem Y is NP-complete if

- Y is in NP and
- $X \leq_{p,Karp} Y$ for every problem X in NP.

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

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

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

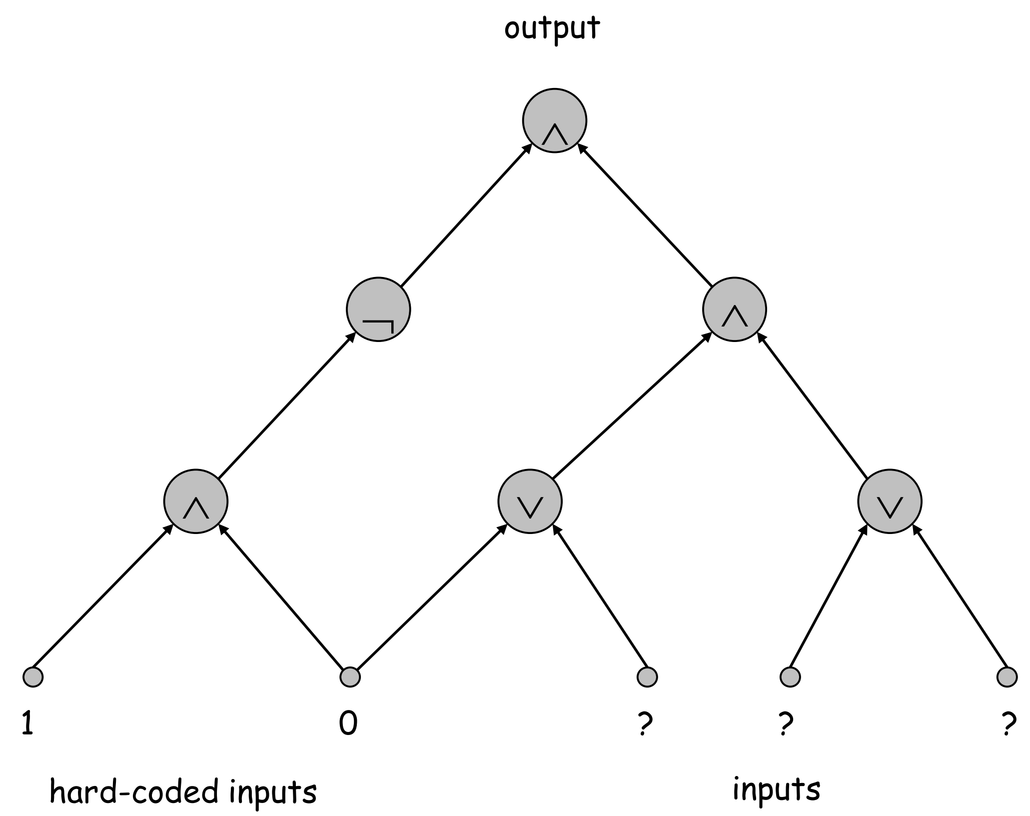
- Let X be any problem in NP. Since $X \leq_{p,Karp} Y$, we can solve X in poly-time. This implies NP $\subseteq$ P.
- We already know P $\subseteq$ NP. Thus P = NP. ▪

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?

yes: 1 0 1



The "First" NP-Complete Problem

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

Proof sketch. CIRCUIT-SAT is in NP (certificate: input on which circuit is 1).

Reduction: **For all X ∈ NP, A $\le_{P,\text{Cook}}$ CIRCUIT-SAT.**

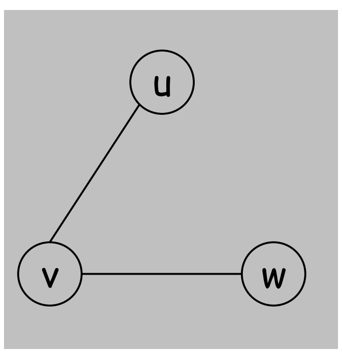
- Any algorithm that takes a fixed number of bits n as input and produces a yes/no answer can be represented by such a circuit.
  Moreover, if algorithm takes poly-time, then circuit is of poly-size.

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

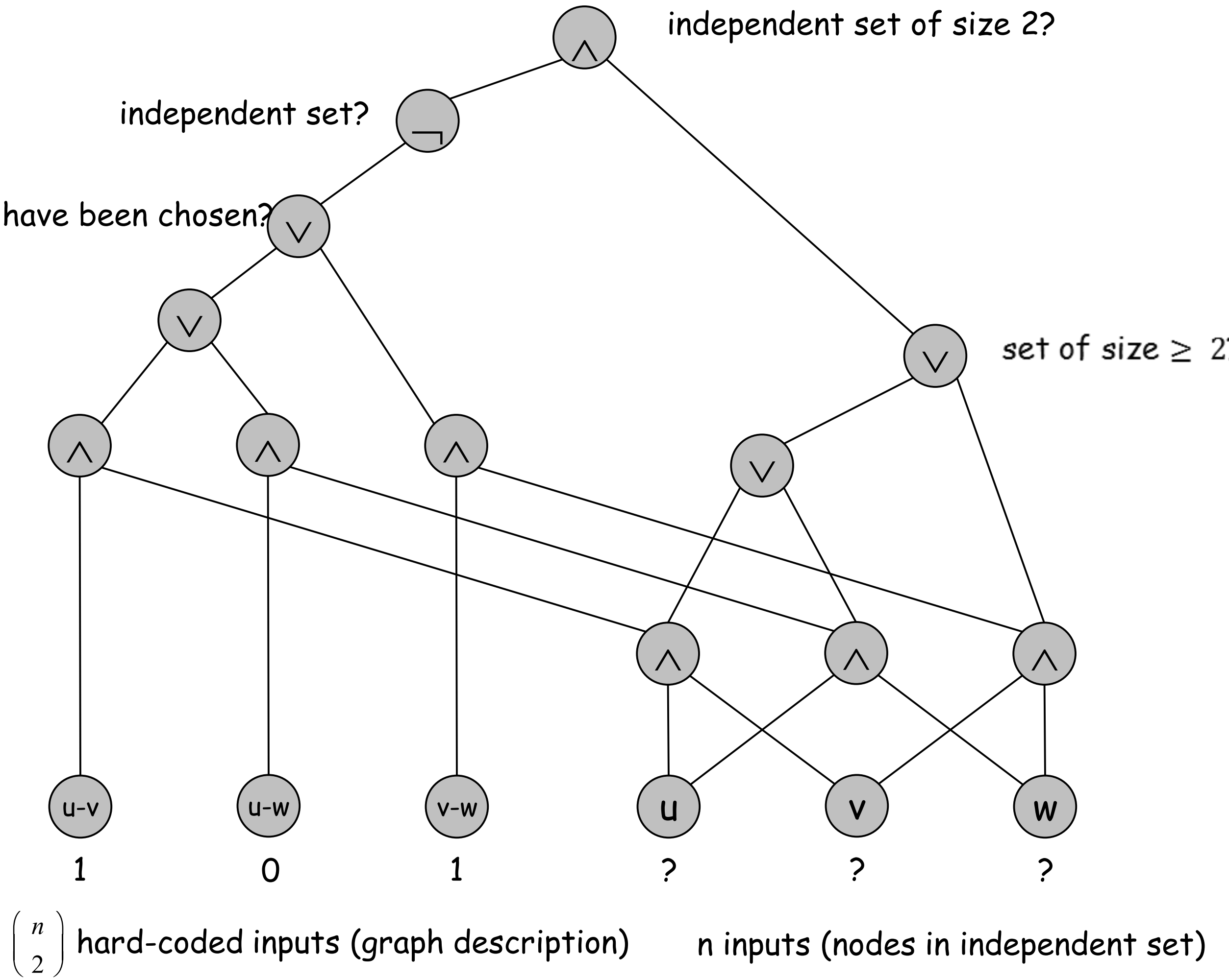
- Since X∈ NP, it has a poly-time certifier C(s, t) that runs in time p(|s|).
  To determine whether s is in X, need to know if there exists a certificate t of length p(|s|) such that C(s, t) = `yes`.
- View C(s, t) as an algorithm on |s| + p(|s|) bits (input s, certificate t) and convert it into a poly-size circuit K.
  - first |s| bits are hard-coded with s
  - remaining p(|s|) bits represent bits of t
- **Correctness:** Circuit K is satisfiable iff C(s, t) = `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



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 $X \leq_{p,\text{Karp}} Y$.

Justification. If X is an NP-complete problem, and Y is a problem in NP with the property that $X \leq_{P,\text{Karp}} Y$ then Y is NP-complete.

Proof. Let W be any problem in NP. Then $W \underbrace{\leq_{P,\text{Karp}}}_{\text{by definition of NP-complete}} X \underbrace{\leq_{P,\text{Karp}}}_{\text{by assumption}} Y$.

- By transitivity, $W \leq_{P,\text{Karp}} Y$.
- Hence Y is NP-complete. ▪

3-SAT is NP-Complete

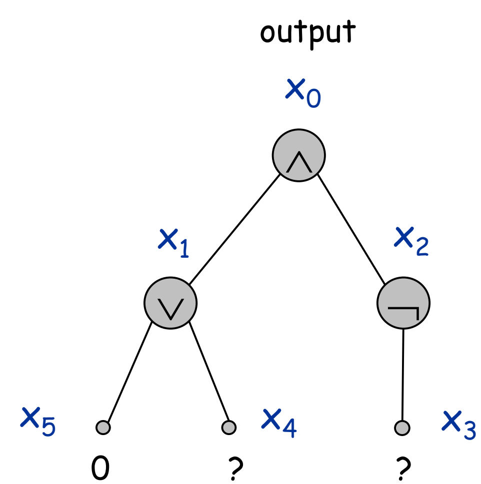
Theorem. 3-SAT is NP-complete.

Proof. Suffices to show that CIRCUIT-SAT $\le_P$ 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: $\overline{x_0} \vee x_1$, $\overline{x_0} \vee x_2$, $x_0 \vee \overline{x_1} \vee \overline{x_2}$

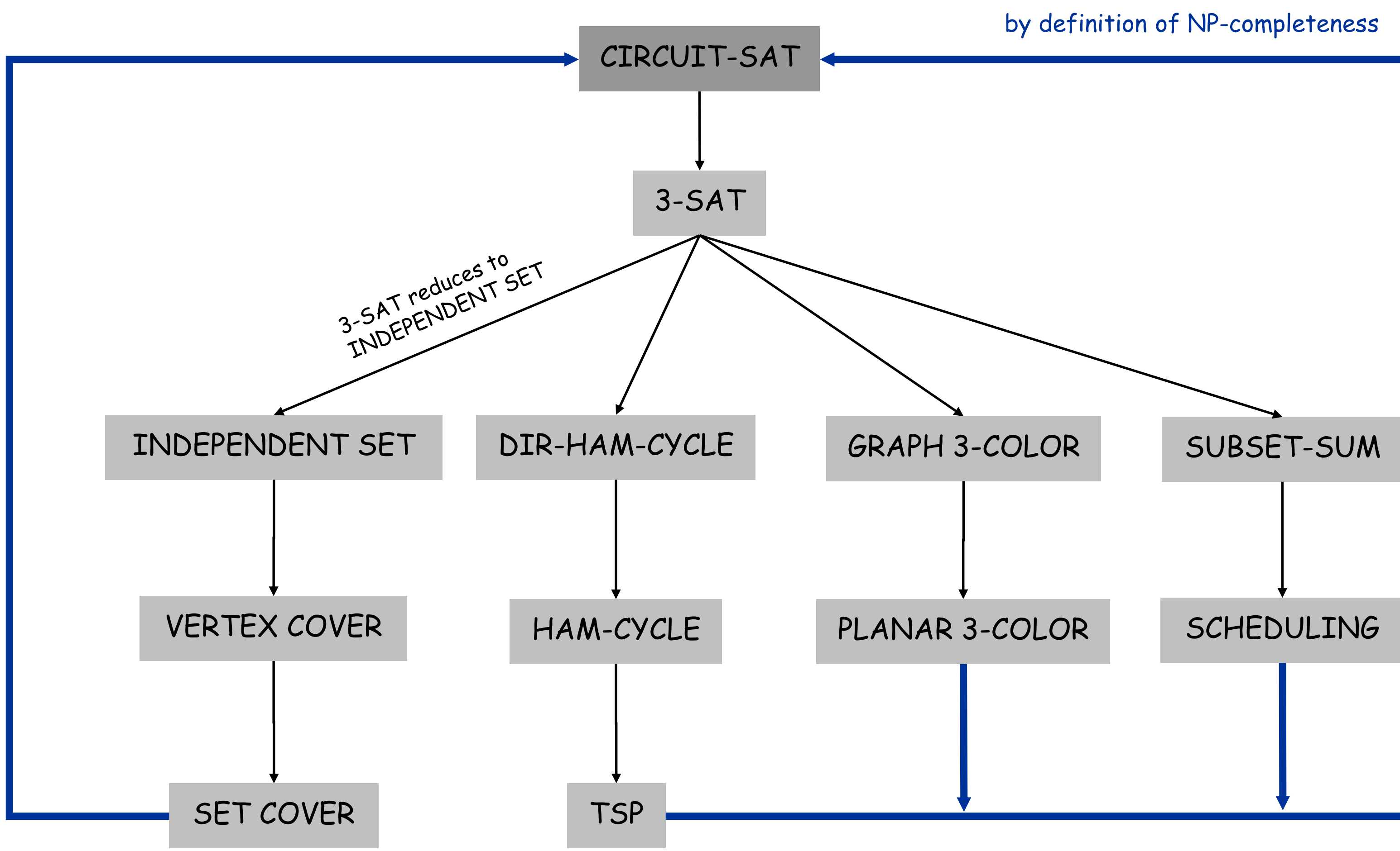
- Hard-coded input values and output value.
  - $x_5 = 0$ $\Rightarrow$ add 1 clause: $\overline{x_5}$
  - $x_0 = 1$ $\Rightarrow$ add 1 clause: x_0

- 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-MATCHIN,G 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]

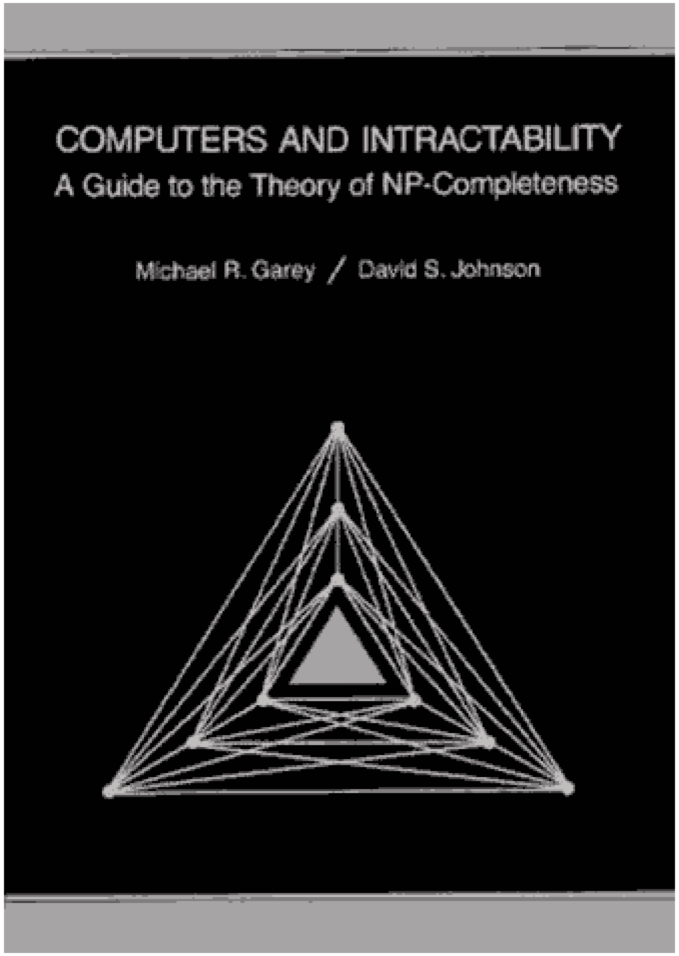
- Prime intellectual export of CS to other disciplines.
- 6,000 citations per year (title, abstract, keywords).
  - more than "compiler", "operating system", "database"
- Broad applicability and classification power.
- "Captures vast domains of computational, scientific, mathematical endeavors, and seems 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 angiocardiogram.
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.


COMPUTERS AND INTRACTABILITY
A Guide to the Theory of NP-Completeness
Michael R. Garey / David S. Johnson

"I can't find an efficient algorithm, but neither can all these famous people."