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) = `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(·) 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. 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	$\begin{bmatrix} 0 & 1 & 1 \\ 2 & 4 & -2 \\ 0 & 3 & 15 \end{bmatrix}, \begin{bmatrix} 4 \\ 2 \\ 36 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$

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 $|t| \le p(|s|)$ 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| ≤ |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.

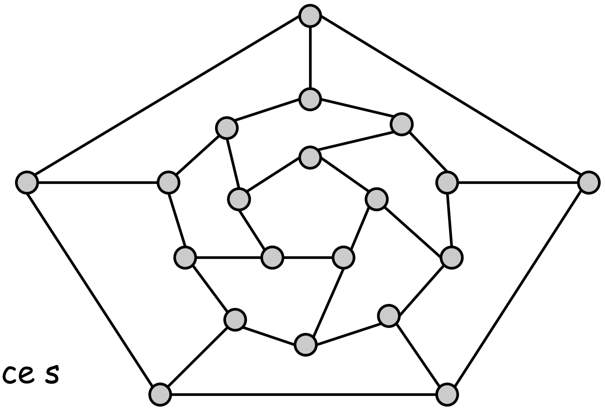
Certifiers and Certificates: Hamiltonian Cycle

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

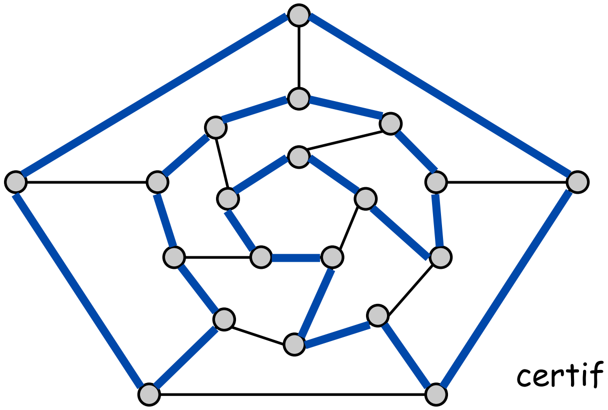
Certificate. A permutation of the n nodes.

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 $C(s, t) = A(s)$. ▪

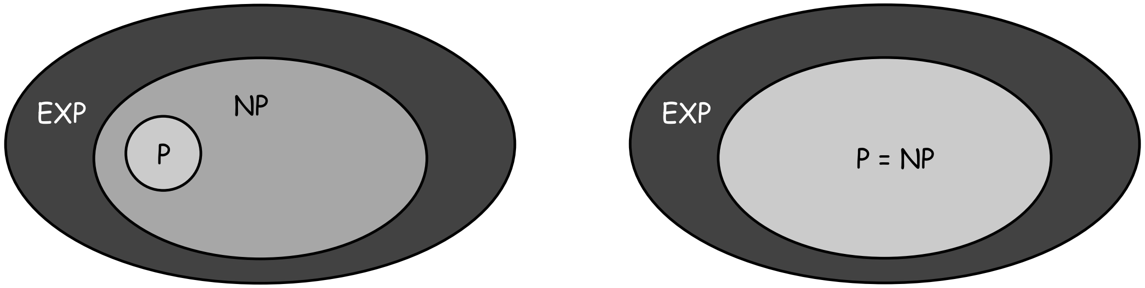
Claim. $NP \subseteq EXP$.
Pf. Consider any problem X in NP.

- By definition, there exists a poly-time certifier $C(s, t)$ for X.
- 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 ≠ 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.

The Simpson's: P = NP?

Futurama: P = NP?

Looking for a Job?

Some writers for the Simpsons and Futurama.

- J. Steward Burns. M.S. in mathematics, Berkeley, 1993.
- David X. Cohen. M.S. in computer science, Berkeley, 1992.
- Al Jean. B.S. in mathematics, Harvard, 1981.
- Ken Keeler. Ph.D. in applied mathematics, Harvard, 1990.
- Jeff Westbrook. Ph.D. in computer science, Princeton, 1989.

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 such that x is a `yes` instance of X iff y is a `yes` instance of Y.

↑

we require |y| to be of 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?

↑

we abuse notation $\le_p$ 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.

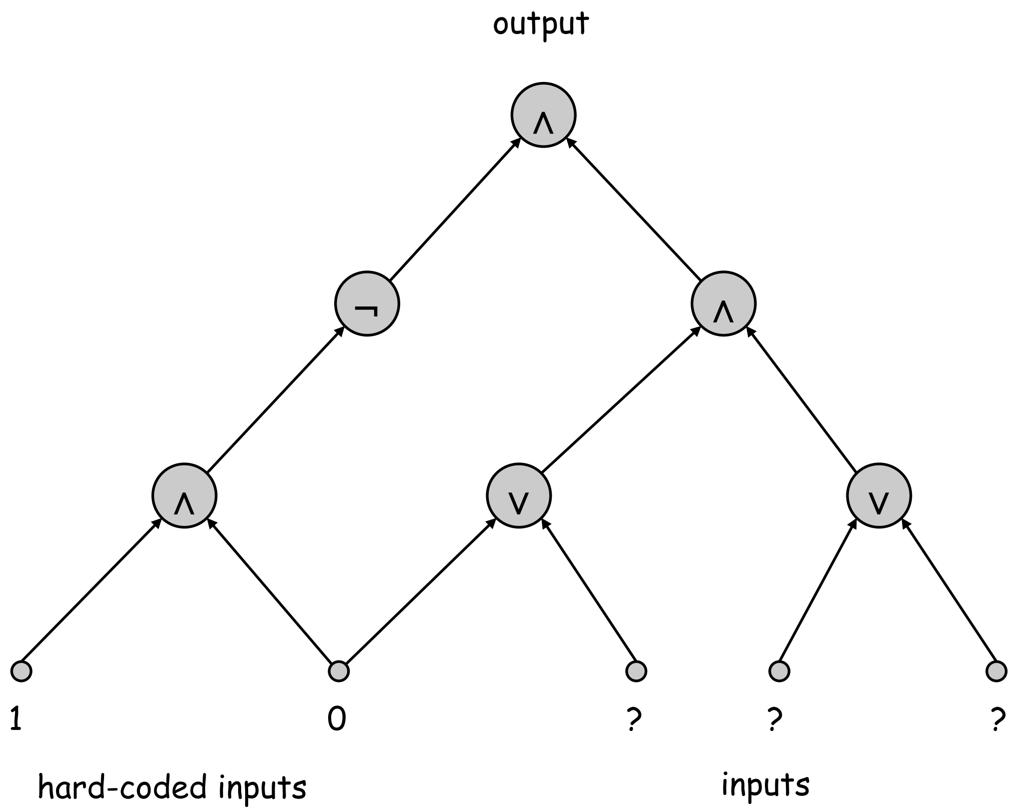
- Let X be any problem in NP. Since $X \leq_p Y$, we can solve X in poly-time. This implies NP $\subseteq$ P.
- We already know P $\subseteq$ NP. Thus P = NP. ▪

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?

yes: 1 0 1



The "First" NP-Complete Problem

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

Pf. (sketch)

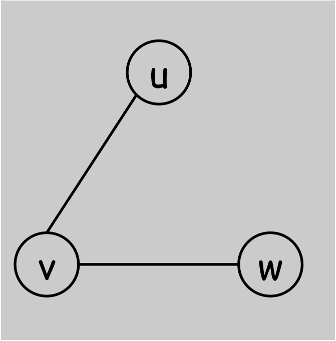
- Any algorithm that takes a fixed number of bits n as input and produces a yes/no answer can be represented by such a circuit. Moreover, if algorithm takes poly-time, then circuit is of poly-size.

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

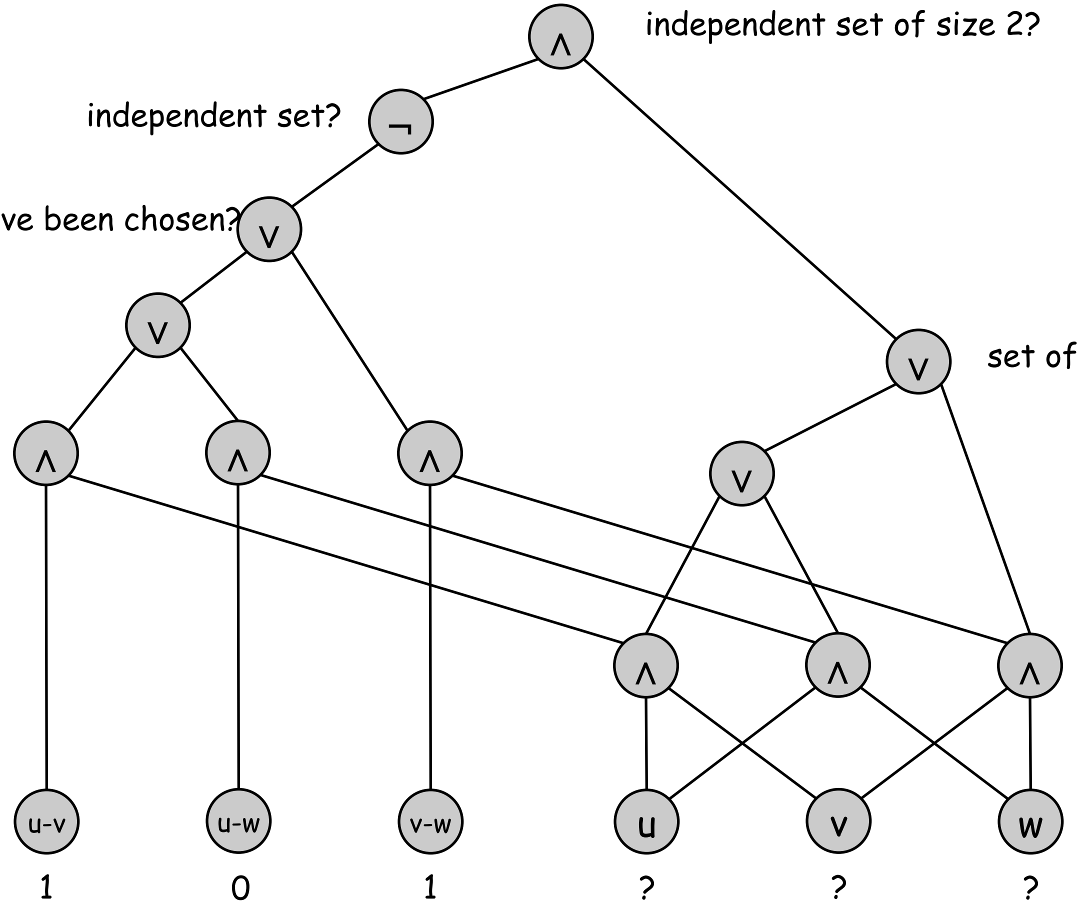
- Consider some problem X in NP. It has a poly-time certifier C(s, t). To determine whether s is in X, need to know if there exists a certificate t of length p(|s|) such that C(s, t) = `yes`.
- View C(s, t) as an algorithm on |s| + p(|s|) bits (input s, certificate t) and convert it into a poly-size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



independent set of size 2?

independent set?

both endpoints of some edge have been chosen?

set of size 2?

$\binom{n}{2}$ 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 an NP-complete problem X.
- Step 3. Prove that $X \le_p Y$.

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

Pf. Let W be any problem in NP. Then $W \underset{\substack{\uparrow \\ \text{by definition of} \\ \text{NP-complete}}}{\le_P} X \underset{\substack{\uparrow \\ \text{by assumption}}}{\le_P} Y.$

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

3-SAT is NP-Complete

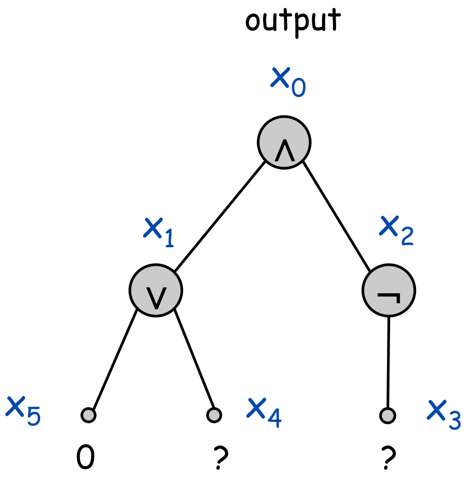
Theorem. 3-SAT is NP-complete.

Pf. 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:
  - $x_2 = \neg\, x_3$ ⇒ add 2 clauses: $x_2 \vee x_3\ ,\ \overline{x_2} \vee \overline{x_3}$
  - $x_1 = x_4 \vee x_5$ ⇒ add 3 clauses: $x_1 \vee \overline{x_4}\,,\ x_1 \vee \overline{x_5}\ ,\ \overline{x_1} \vee x_4 \vee x_5$
  - $x_0 = x_1 \wedge x_2$ ⇒ add 3 clauses: $\overline{x_0} \vee x_1\,,\ \overline{x_0} \vee x_2\,,\ x_0 \vee \overline{x_1} \vee \overline{x_2}$

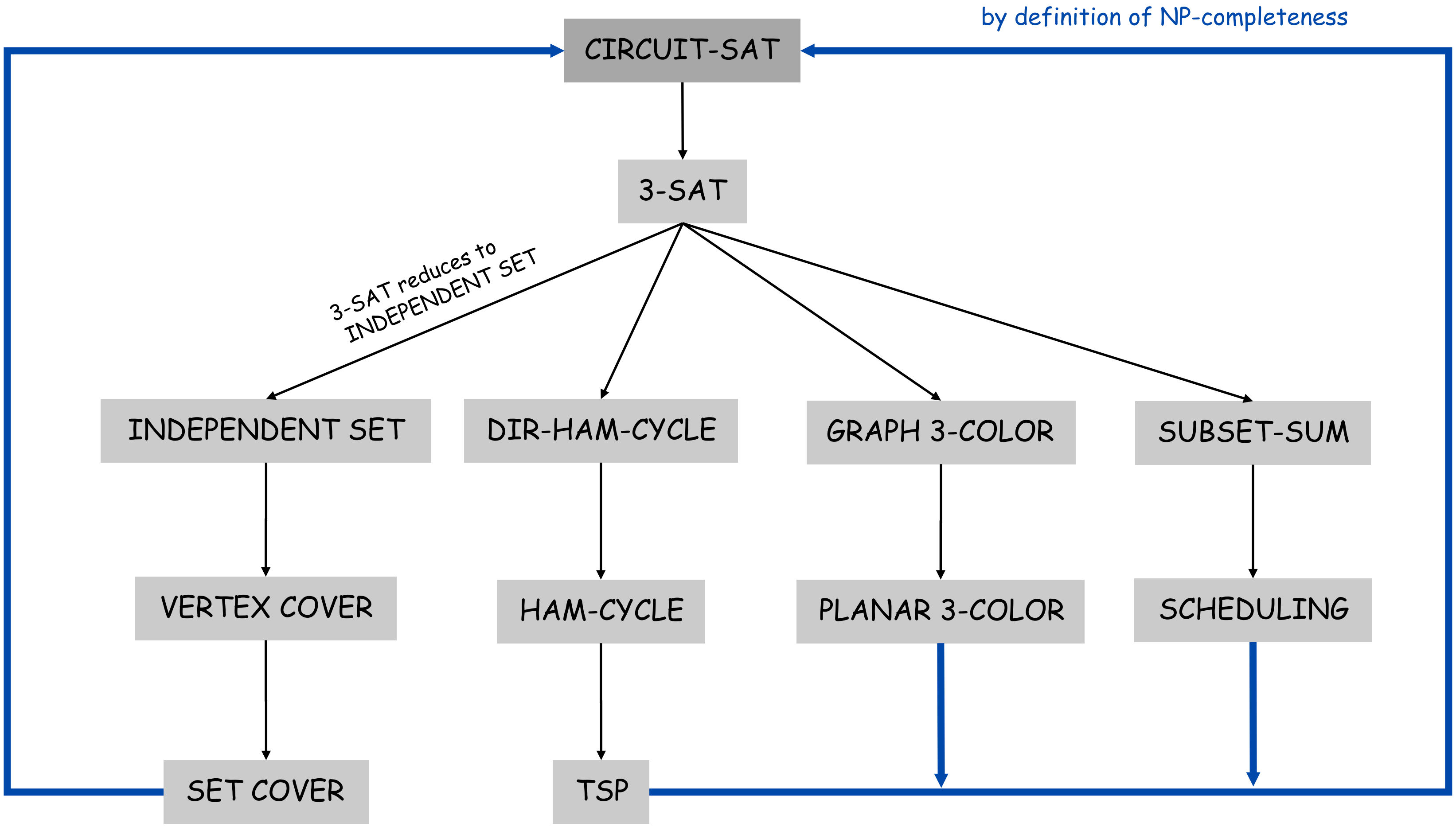
- Hard-coded input values and output value.
  - $x_5 = 0$ ⇒ add 1 clause: $\overline{x_5}$
  - $x_0 = 1$ ⇒ add 1 clause: x_0

- 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.