Algorithm Design and Analysis (H)

CS216

Instructor: Shan CHEN (陈杉)

chens3@sustech.edu.cn

(slides edited from Prof. Shiqi Yu)

南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

NP and Computational Intractability

5. **P** vs. **NP**

P

- **Decision problem.**
  - Problem X is a set of strings.
  - Instance s is one string.
  - Algorithm A solves problem X: A(s) = *yes* if and only if $s \in X$.
- **Def.** Algorithm A runs in polynomial time if for every string s, A(s) terminates in $\leq p(|s|)$ "steps", where $p(\cdot)$ is some polynomial function.

length of s

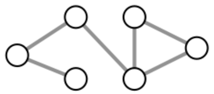
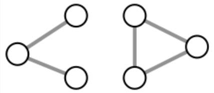
- **P.** Set of decision problems for which there exists a poly-time algorithm.

on a deterministic Turing machine

problem PRIMES:	{ 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, ... }
instance s:	592335744548702854681
algorithm:	Agrawal–Kayal–Saxena (2002)

Some Problems in **P**

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

problem	description	poly-time algorithm	yes	no
MULTIPLE	Is x a multiple of y ?	grade-school division	51, 17	51, 16
REL-PRIME	Are x and y relatively prime ?	Euclid's algorithm	34, 39	34, 51
PRIMES	Is x prime ?	Agrawal–Kayal–Saxena	53	51
EDIT-DISTANCE	Is the edit distance between x and y less than 5 ?	Needleman–Wunsch	niether neither	acgggt ttttta
L-SOLVE	Is there a vector x that satisfies $Ax = b$?	Gauss–Edmonds elimination	$\begin{bmatrix} 0 & 1 & 1 \\ 2 & 4 & -2 \\ 0 & 3 & 15 \end{bmatrix}, \begin{bmatrix} 4 \\ 2 \\ 36 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$
U-CONN	Is an undirected graph G connected?	depth-first search		

NP

- **Certification 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 shows $s \in X$.
- **Def.** Algorithm $C(s, t)$ is a certifier for problem X if for every string s, $s \in X$ if and only if there exists a string t such that $C(s, t) =$ *yes*.
- **NP.** Set of decision problems for which there exists a poly-time certifier.
  - $C(s, t)$ is a poly-time algorithm.
  - Certificate t is of polynomial size: $|t| \leq p(|s|)$ for some polynomial $p(\cdot)$.

also called "witness" or "proof"

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

Certifiers and Certificates: Composites

- **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.** Grade-school division.

instance s:	437669	
certificate t:	541	← $437{,}669 = 541 \times 809$

- **Conclusion.** COMPOSITES $\in$ **NP**.

Certifiers and Certificates: Satisfiability

- **SAT.** Given a CNF formula Φ, is there a satisfying truth assignment?
- **3-SAT.** SAT where each clause contains exactly 3 literals.

- **Certificate.** An assignment of truth values to the Boolean variables.
- **Certifier.** Check that each clause in Φ has at least one true literal.

instance s	$\Phi = \left(\overline{x_1} \vee x_2 \vee x_3\right) \wedge \left(x_1 \vee \overline{x_2} \vee x_3\right) \wedge \left(\overline{x_1} \vee x_2 \vee x_4\right)$
certificate t	$x_1 = \mathit{true},\; x_2 = \mathit{true},\; x_3 = \mathit{false},\; x_4 = \mathit{false}$

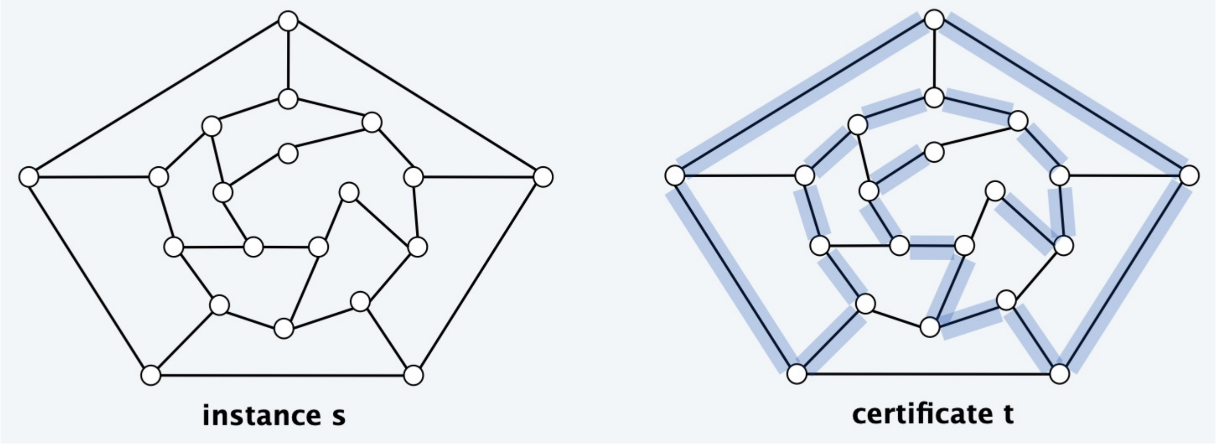
- **Conclusions.** SAT $\in$ **NP**, 3-SAT $\in$ **NP**.

Certifiers and Certificates: Hamiltonian Path

- **HAMILTONIAN-PATH.** Given an undirected graph G = (V, E), does there exist a simple path that visits every node?

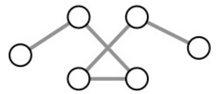
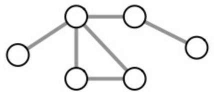
- **Certificate.** A permutation π of the *n* nodes.
- **Certifier.** Check that π contains each node in V exactly once, and that G contains an edge between each pair of adjacent nodes in π.

- **Conclusion.**
  HAMILTONIAN-PATH ∈ **NP**.



Some Problems in **NP**

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

problem	description	poly-time algorithm	yes	no
L-SOLVE	Is there a vector x that satisfies $Ax = b$?	Gauss–Edmonds elimination	$\begin{bmatrix} 0 & 1 & 1 \\ 2 & 4 & -2 \\ 0 & 3 & 15 \end{bmatrix}, \begin{bmatrix} 4 \\ 2 \\ 36 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$
COMPOSITES	Is x composite?	Agrawal–Kayal–Saxena	51	53
FACTOR	Does x have a nontrivial factor less than y?	???	(56159, 50)	(55687, 50)
SAT	Given a CNF formula, does it have a satisfying truth assignment?	???	$\neg x_1 \lor x_2 \lor \neg x_3$ $x_1 \lor \neg x_2 \lor x_3$ $\neg x_1 \lor \neg x_2 \lor x_3$	$\neg x_2$ $x_1 \lor x_2$ $\neg x_1 \lor x_2$
HAMILTON-PATH	Is there a simple path between u and v that visits every node?	???		

Significance of **NP**

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

> " NP captures vast domains of computational, scientific, and mathematical endeavors, and seems to roughly delimit what mathematicians and scientists have been aspiring to compute feasibly. " — Christos Papadimitriou

P, NP, and EXP

- **P.** Decision problems for which there is a poly-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 = ε (empty string), certifier C(s, t) = A(s). ▪

P, NP, and EXP

- **NP.** Decision problems for which there is a poly-time certifier.
- **EXP.** Decision problems for which there is an exponential-time algorithm.

- **Claim.** **NP** $\subseteq$ **EXP**.
- **Pf.** Consider any problem $X \in$ **NP**.
  - By definition, there exists a poly-time certifier $C(s, t)$ for X, where certificate t satisfies $|t| \leq p(|s|)$ for some polynomial $p(\cdot)$.
  - To solve instance 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 potential certificates. ▪

- **Fact.** **P** $\neq$ **EXP** $\Rightarrow$ either **P** $\neq$ **NP**, or **NP** $\neq$ **EXP**, or both.

The Main Question: **P** vs. **NP**

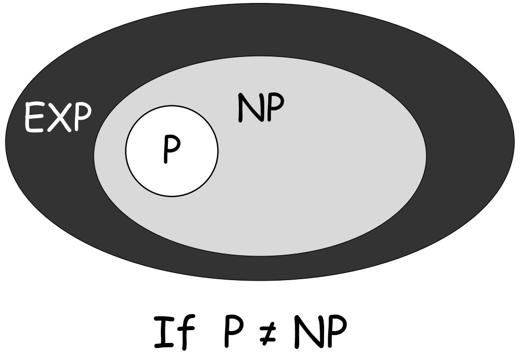
- **Q.** How to solve an instance of 3-SAT with *n* variables?
- **A.** Exhaustive search: try all 2^n truth assignments.

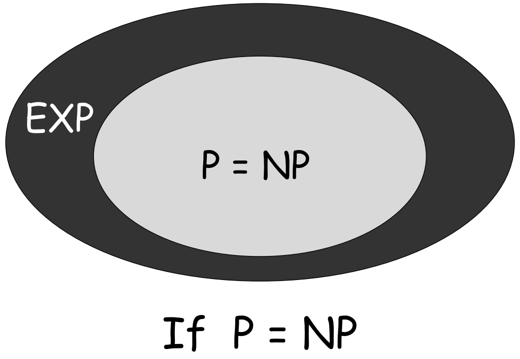
- **Q.** Can we do anything substantially more clever?
- **Conjecture.** No poly-time algorithm for 3-SAT.

intractability

The Main Question: **P** vs. **NP**

- **Does P = NP?** [Cook 1971, Edmonds, Levin, Yablonski, Gödel]
  - Is the decision problem as easy as the certification problem?
  - Millennium prize problem by Clay Mathematics Institute: $1 million





break RSA cryptosystem and potentially crash economy

- **If yes...** Efficient algorithms for 3-SAT, TSP, VERTEX-COVER, FACTOR, ...
- **If no...** No efficient algorithms possible for 3-SAT, TSP, VERTEX-COVER, ...
- **Consensus opinion.** Probably no.

Possible Outcomes: P ≠ NP

" I conjecture that there is no good algorithm for the traveling salesman problem. My reasons are the same as for any mathematical conjecture: (i) It is a legitimate mathematical possibility and (ii) I do not know." — Jack Edmonds (1966)

" In my view, there is no way to even make intelligent guesses about the answer to any of these questions. If I had to bet now, I would bet that P is not equal to NP. I estimate the half-life of this problem at 25–50 more years, but I wouldn't bet on it being solved before 2100. " — Bob Tarjan (2002)

" We seem to be missing even the most basic understanding of the nature of its difficulty... All approaches tried so far probably (in some cases, provably) have failed. In this sense P = NP is different from many other major mathematical problems on which a gradual progress was being constantly done (sometimes for centuries) whereupon they yielded, either completely or partially. " — Alexander Razborov (2002)

Possible Outcomes: **P = NP**

> " I think that in this respect I am on the loony fringe of the mathematical community: I think (not too strongly!) that P = NP and this will be proved within twenty years. Some years ago, Charles Read and I worked on it quite bit, and we even had a celebratory dinner in a good restaurant before we found an absolutely fatal mistake. " — Béla Bollobás (2002)

> " In my opinion this shouldn't really be a hard problem; it's just that we came late to this theory, and haven't yet developed any techniques for proving computations to be hard. Eventually, it will just be a footnote in the books. " — John Conway

Other Possible Outcomes

- **P** = **NP**, but only $\Omega(n^{100})$ algorithm for 3-SAT.
- **P** ≠ **NP**, but with $O(n^{\log^* n})$ algorithm for 3-SAT.
- **P** ≠ **NP** is independent (of ZFC axiomatic set theory).

> " It will be solved by either 2048 or 4096. I am currently somewhat pessimistic. The outcome will be the truly worst case scenario: namely that someone will prove P = NP because there are only finitely many obstructions to the opposite hypothesis; hence there exists a polynomial time solution to SAT but we will never know its complexity! "
>
> — Donald Knuth

6. **NP**-Complete

Polynomial Transformations

- **Def.** Problem X polynomial (Cook) reduces to problem Y if arbitrary instances of problem X can be solved using:

  we require |y| to be of size polynomial in |x|

  - Polynomial number of standard computational steps, plus
  - Polynomial number of calls to oracle that solves problem Y.
- **Def.** Problem X polynomial (Karp) transforms to problem Y if given any instance x of X, we can construct an instance y of 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 with respect to **NP**?

we abuse notation $\le_p$ and blur distinction

NP-Complete and **NP**-hard

- **NP-complete.** Set of problems 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 $Y \in$ **P** iff **P** = **NP**.
- **Pf.** $\Leftarrow$: If **P** = **NP** then $Y \in$ **P** because $Y \in$ **NP**.

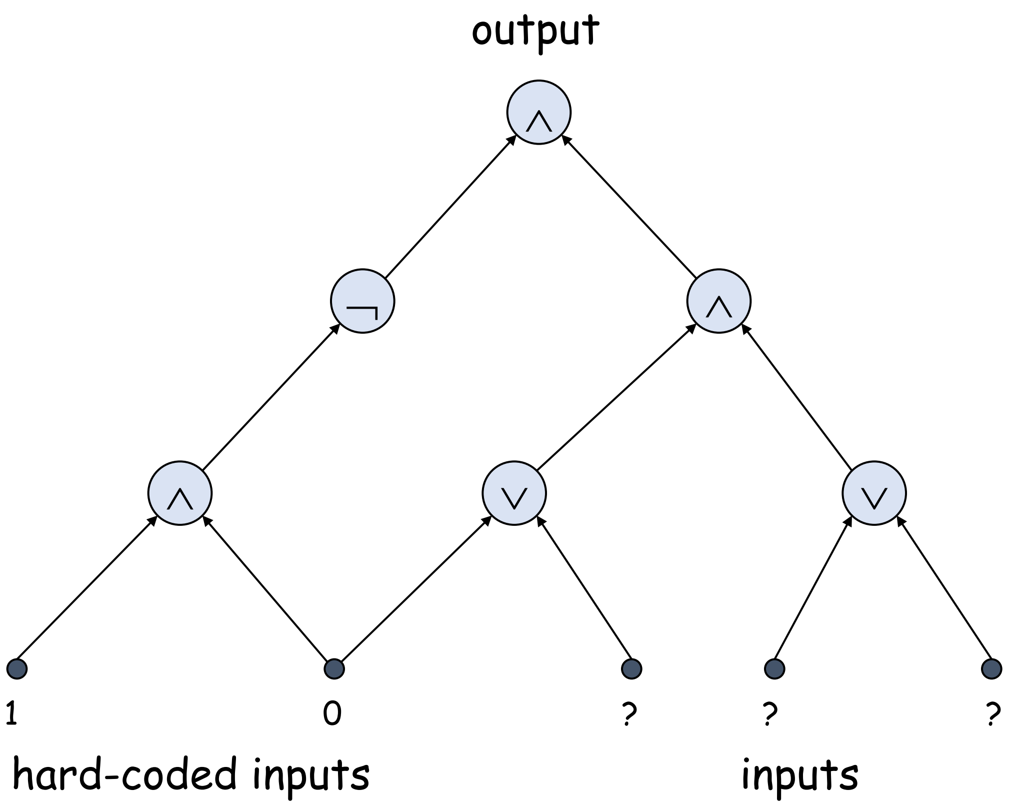
  $\Rightarrow$: Suppose $Y \in$ **P**.
  - Consider any problem $X \in$ **NP**. Since $X \leq_p Y$, we have $X \in$ **P**. This implies **NP** $\subseteq$ **P**.
  - We already know **P** $\subseteq$ **NP**. Thus **P** = **NP**. ▪

- **NP-hard.** [Bell Labs, Steve Cook, Ron Rivest, Sartaj Sahni]
  Set of problems such that every problem in **NP** poly-time reduces to it.
- **Fundamental question.** Are there any "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

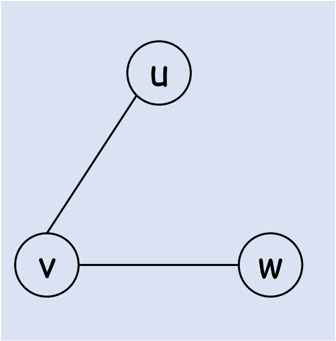
- **Theorem.** CIRCUIT-SAT is **NP**-complete. [Cook 1971, Levin 1973]
- **Pf idea.**

basic distinction between algorithms and circuits

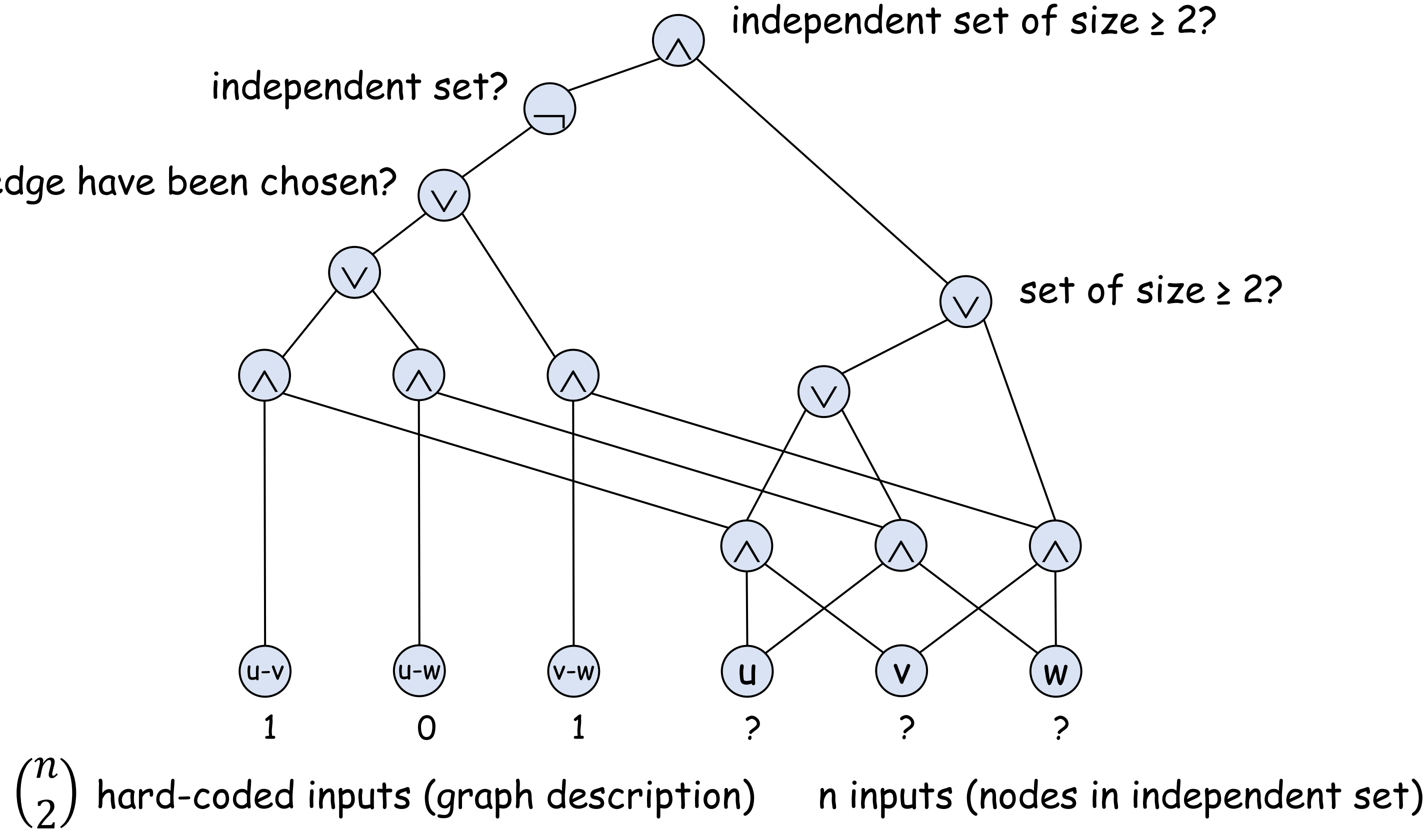
  - Any deterministic algorithm that takes a fixed number of bits as input and produces a yes/no answer can be represented by such a circuit. Moreover, if algorithm runs in poly-time, then circuit is of poly-size.
  - Consider any problem $X \in$ **NP**. It has a poly-time certifier C(s, t). To determine whether $s \in X$, one needs 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*.

CIRCUIT-SAT Transformation Example

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



G = (V, E), n = 3



Establishing **NP**-Completeness

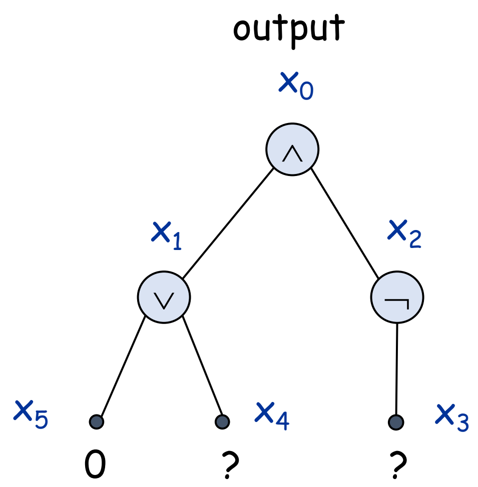
- **Remark.** Once we establish first "natural" **NP**-complete problem, others fall like dominoes.
- **Recipe.** To prove that $Y \in$ **NP**-complete:
  - Step 1. Show that Y is in **NP**.
  - Step 2. Choose an **NP**-complete problem X.
  - Step 3. Prove that $X \leq_p Y$.
- **Claim.** If $X \in$ **NP**-complete, $Y \in$ **NP**, and $X \leq_P Y$, then $Y \in$ **NP**-complete.
- **Pf.** Consider any problem $W \in$ **NP**. Then $W \leq_P X \leq_P Y$.
  - By transitivity, $W \leq_P Y$.
  - Hence $Y \in$ **NP**-complete. ▪

by definition of NP-complete

by assumption

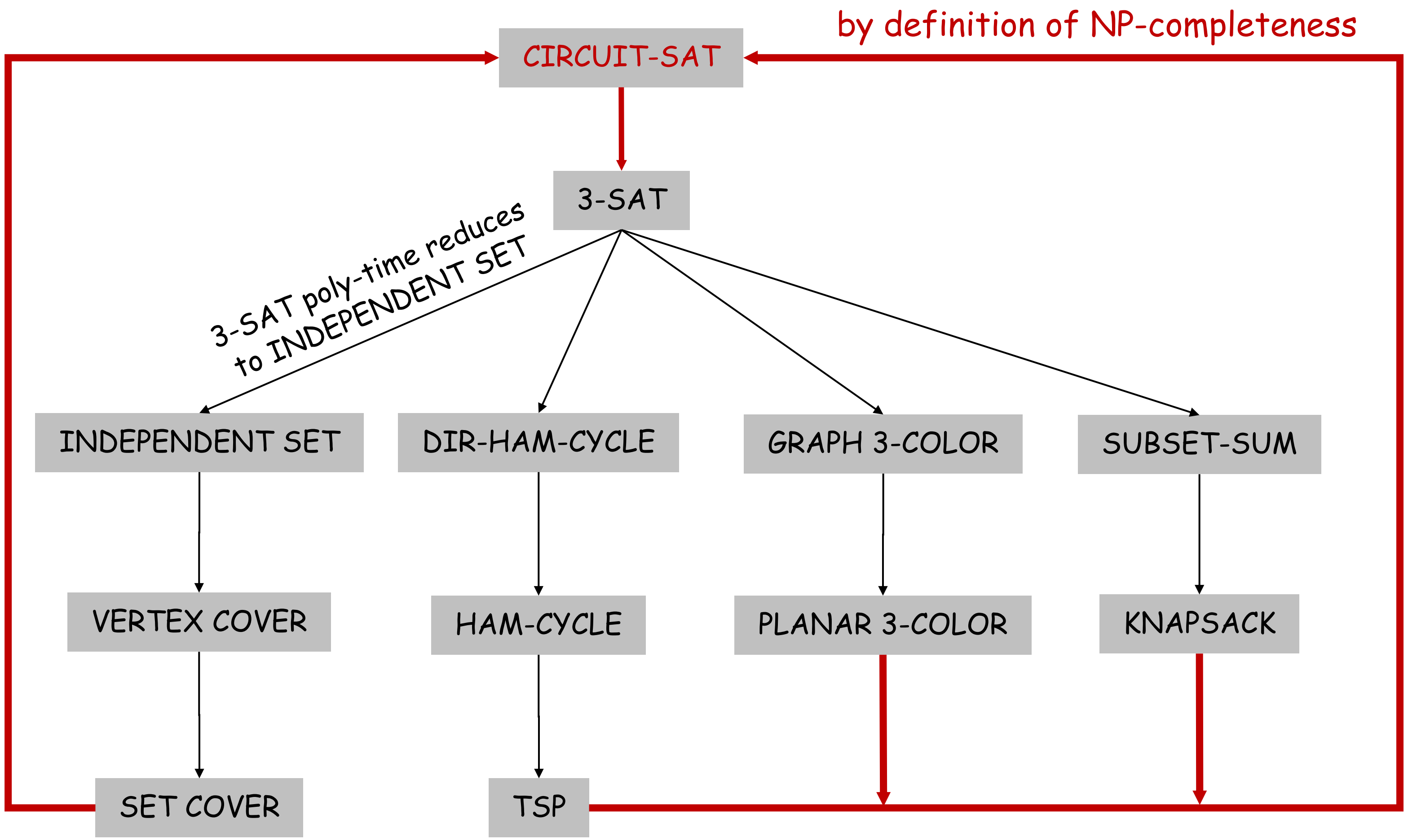
3-SAT is **NP**-Complete

- **Theorem.** 3-SAT is **NP**-complete.
- **Pf.** Suffices to show that CIRCUIT-SAT $\leq_P$ 3-SAT since 3-SAT $\in$ **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: $\overline{x_0} \vee x_1$, $\overline{x_0} \vee x_2$, $x_0 \vee \overline{x_1} \vee \overline{x_2}$
  - Output value and hard-coded input values:
    - ✓ $x_5 = 0 \Rightarrow$ 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.
    - ✓ Add extra variables z_1, z_2 such that they have to be 0 to satisfy 3-SAT. ▪



Implications of Karp and Cook-Levin

- **Observation.** All of these problems are **NP**-complete, i.e., of the same hardness.



Some **NP**-Complete Problems

- **Basic genres of NP-complete problems and paradigmatic examples.**
  - Packing/covering problems: SET-COVER, VERTEX-COVER, INDEPENDENT-SET.
  - Constraint satisfaction problems: CIRCUIT-SAT, SAT, 3-SAT.
  - Sequencing problems: HAMILTONIAN-PATH, HAMILTONIAN-CYCLE, TSP.
  - Partitioning problems: 3D-MATCHING, 3-COLOR.
  - Numerical problems: SUBSET-SUM, KNAPSACK.
- **Practice**. Most **NP** problems are known to be either in **P** or **NP**-complete.
- **Notable exceptions.** FACTOR, DISCRETE-LOG, GRAPH-ISOMORPHISM, …
- **Theorem.** [Ladner 1975] Unless **P = NP**, there exist problems in **NP** that are neither in **P** nor **NP**-complete.

More Hard Computational Problems

Aerospace engineering. Optimal mesh partitioning for finite elements.

Biology. Phylogeny reconstruction.

Chemical engineering. Heat exchanger network synthesis.

Chemistry. Protein folding.

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. Minimum risk portfolio of given return.

Game theory. Nash equilibrium that maximizes social welfare.

Mathematics. Given integer a_1, …, a_n, compute $\int_0^{2\pi} \cos(a_1\theta) \times \cos(a_2\theta) \times \cdots \times \cos(a_n\theta)\, d\theta$

Mechanical engineering. Structure of turbulence in sheared flows.

Medicine. Reconstructing 3d shape from biplane angiocardiogram.

Operations research. Traveling salesperson problem.

Physics. Partition function of 3d Ising model.

Politics. Shapley–Shubik voting power.

Recreation. Versions of Sudoku, Checkers, Minesweeper, Tetris, Rubik's Cube.

Statistics. Optimal experimental design.

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 (more than "compiler", "OS", "database").
  - Broad applicability and classification power.

- **NP-completeness can guide scientific inquiry.**
  - 1926: Ising introduces simple model for phase transitions.
  - 1944: Onsager finds closed-form solution to 2D-ISING in tour de force.
  - 19xx: Feynman and other top minds seek solution to 3D-ISING.
  - 2000: Istrail proves 3D-ISING $\in$ **NP**-complete.

a holy grail of statistical mechanics

search for closed formula appears doomed

Exploiting Intractability

- **Q.** Is FACTOR $\in$ **P**?
- **A.** Unknown.

- **Challenge.** Factor this number:

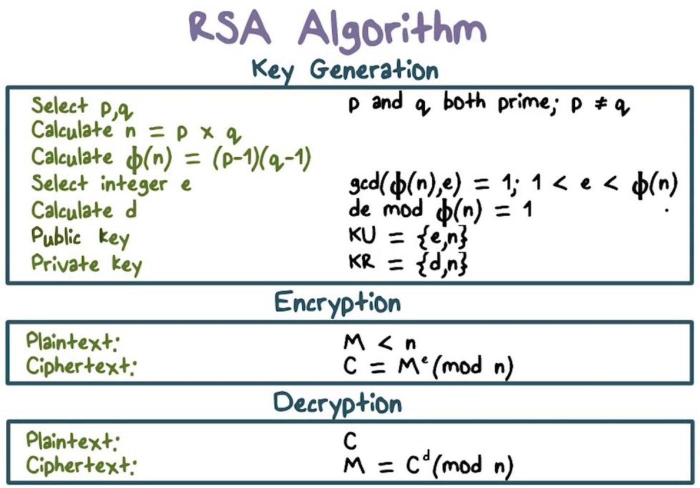
74037563479561712828046796097429573142593188889231289
08493623263897276503402826627689199641962511784399589
43305021275853701189680982867331732731089309005525051
16877063299072396380786710086096962537934650563796359

RSA-704
($30,000 prize if you can factor)

Exploiting Intractability

- **Modern cryptography.**
  - Ex. Send your credit card number to Amazon.
  - Ex. Digitally sign an e-document.
  - Enables freedom of privacy, speech, press, political association.
- **RSA.** Based on dichotomy between complexity of two problems.
  - To use: generate two random *n*-bit primes and multiply.
  - To break: suffices to factor a 2*n*-bit integer.

Sold for $2.1 billion



Exploiting Intractability

- **Modern cryptography.**
  - Ex. Send your credit card number to Amazon.
  - Ex. Digitally sign an e-document.
  - Enables freedom of privacy, speech, press, political association.
- **RSA.** Based on dichotomy between complexity of two problems.
  - To use: generate two random n-bit primes and multiply.
  - To break: suffices to factor a $2n$-bit integer.
- **Theorem.** [Shor 1994] Can factor an n-bit integer in $O(n^3)$ steps on a quantum computer.
  - 15 = 3 x 5 factored in 2001; 21 = 3 x 7 factored in 2012.
- **Fundamental question.** Does **P** = **BQP**?

quantum analog of P
(bounded error quantum polynomial time)