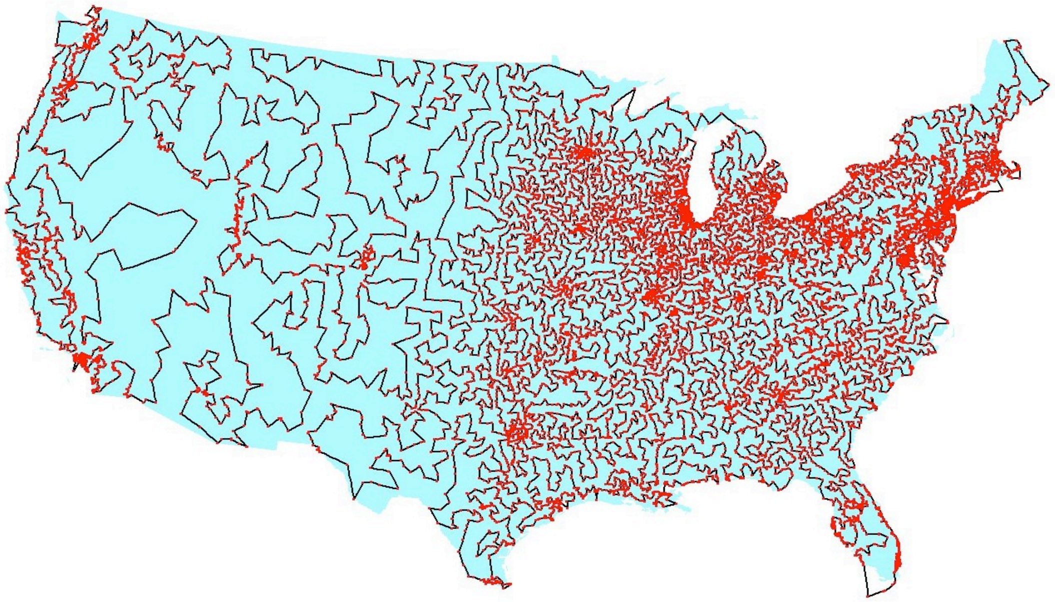
Intractability

A difficult problem

Traveling salesperson problem (TSP)

Given: A set of N cities and $M for gas.

Problem: Does a traveling salesperson have enough $ for gas to visit all the cities?



An algorithm ("exhaustive search"):

Try all N! orderings of the cities to find one that can be visited for $M

A Reasonable Question about Algorithms

Q. Which algorithms are useful in practice?

A. [von Neumann 1953, Gödel 1956, Cobham 1964, Edmonds 1965, Rabin 1966]

- Model of computation = deterministic Turing machine.
- Measure running time as a function of input size N.
- Polynomial time: Number of steps less than aN^b for some constants a, b.
- Useful in practice ("efficient") = polynomial time for all inputs.

Ex 1. Sorting N elements

Insertion sort takes less than aN^2 steps for all inputs.

efficient

Ex 2. TSP on N cities

Exhaustive search could take aN! steps.

not efficient

In theory: Definition is broad and robust (since a and b tend to be small).

In practice: Poly-time algorithms tend to scale to handle large problems.

Exponential Growth

Exponential growth dwarfs technological change.

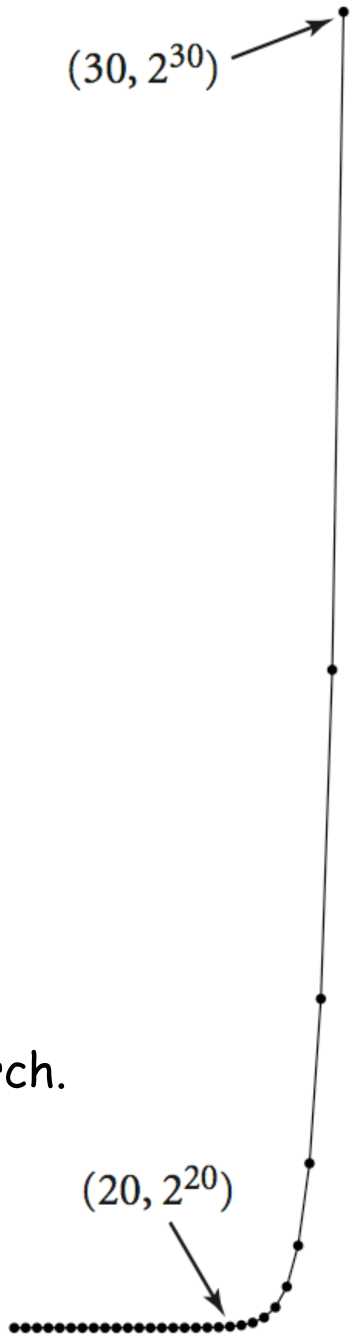
- Suppose you have a giant parallel computing device…
- With as many processors as electrons in the universe…
- And each processor has power of today's supercomputers…
- And each processor works for the life of the universe…

quantity	value
electrons in universe †	10^{79}
supercomputer instructions per second	10^{13}
age of universe in seconds †	10^{17}

† estimated

- Will not help solve 1,000 city TSP problem via exhaustive search.

$1000! \gg 10^{1000} \gg 10^{79} \times 10^{13} \times 10^{17}$



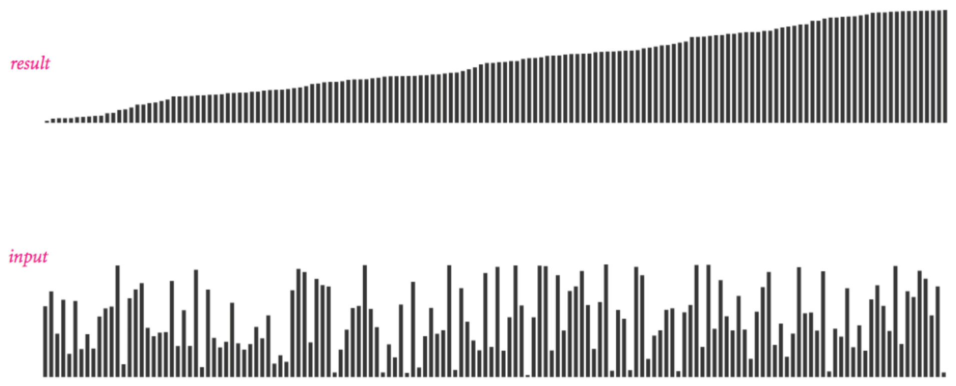
Reasonable Questions about Problems

Q. Which problems can we solve in practice?

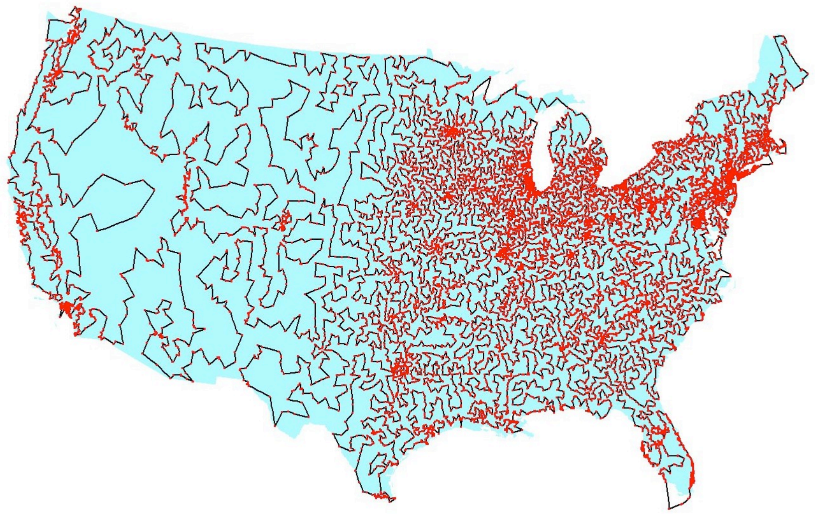
A. Those with easy-to-find answers or with guaranteed poly-time algorithms.

Q. Which problems have guaranteed poly-time algorithms?

A. Not so easy to know. Focus of today's lecture.



many known poly-time algorithms for sorting



no known poly-time algorithm for TSP

Four Fundamental Problems

LSOLVE. Given a system of linear equations, find a solution.

$$\begin{array}{rcrcrcr} 0x_0 & + & 1x_1 & + & 1x_2 & = & 4 \\ 2x_0 & + & 4x_1 & - & 2x_2 & = & 2 \\ 0x_0 & + & 3x_1 & + & 15x_2 & = & 36 \end{array} \qquad \begin{array}{rcr} x_0 & = & -1 \\ x_1 & = & 2 \\ x_2 & = & 2 \end{array}$$

$\longleftarrow$ variables are real numbers

LP. Given a system of linear inequalities, find a solution.

$$\begin{array}{rcrcrcr} 48x_0 & + & 16x_1 & + & 119x_2 & \le & 88 \\ 5x_0 & + & 4x_1 & + & 35x_2 & \ge & 13 \\ 15x_0 & + & 4x_1 & + & 20x_2 & \ge & 23 \\ x_0 & , & x_1 & , & x_2 & \ge & 0 \end{array} \qquad \begin{array}{rcr} x_0 & = & 1 \\ x_1 & = & 1 \\ x_2 & = & 1/5 \end{array}$$

$\longleftarrow$ variables are real numbers

ILP. Given a system of linear inequalities, find a 0-1 solution.

$$\begin{array}{rcrcrcr} & & x_1 & + & x_2 & \ge & 1 \\ x_0 & & & + & x_2 & \ge & 1 \\ x_0 & + & x_1 & + & x_2 & \le & 2 \end{array} \qquad \begin{array}{rcr} x_0 & = & 0 \\ x_1 & = & 1 \\ x_2 & = & 1 \end{array}$$

$\longleftarrow$ variables are 0 or 1

SAT. Given a system of boolean equations, find a solution.

$$\begin{array}{llllll} (x_0 \text{ and } x_1 \text{ and } x_2) & \text{or} & (x_1 \text{ and } x_2) & \text{or} & (x_0 \text{ and } x_2) & = \text{true} \\ (x_0 \text{ and } x_1) & & & \text{or} & (x_1 \text{ and } x_2) & = \text{false} \\ (x_1 \text{ and } x_2) & \text{or} & (x_0 \text{ and } x_2) & \text{or} & (x_0) & = \text{true} \end{array} \qquad \begin{array}{ll} x_0 & = \text{false} \\ x_1 & = \text{true} \\ x_2 & = \text{true} \end{array}$$

$\longleftarrow$ variables are "true" or "false"

Four Fundamental Problems

LSOLVE. Given a system of linear equations, find a solution.

LP. Given a system of linear inequalities, find a solution.

ILP. Given a system of linear inequalities, find a binary solution.

SAT. Given a system of boolean equations, find a solution.

Q. Which of these problems have guaranteed poly-time solutions?

A. No easy answers.

✓ LSOLVE. Yes. Gaussian elimination solves n-by-n system in n^3 time.

✓ LP. Yes. Ellipsoid algorithm is poly-time. ← problem was open for decades

? ILP, SAT. No poly-time algorithm known or believed to exist!

Search Problems

or report none exists

Search problem. Given an instance I of a problem, find a solution S.

Requirement. Must be able to efficiently check that S is a solution.

poly-time in size of instance I

Search Problems

or report none exists

Search problem. Given an instance I of a problem, find a solution S.

Requirement. Must be able to efficiently check that S is a solution.

poly-time in size of instance I

LSOLVE. Given a system of linear equations, find a solution.

$$\begin{array}{rcrcrcr} 0x_0 & + & 1x_1 & + & 1x_2 & = & 4 \\ 2x_0 & + & 4x_1 & - & 2x_2 & = & 2 \\ 0x_0 & + & 3x_1 & + & 15x_2 & = & 36 \end{array}$$

instance I

$$\begin{array}{rcr} x_0 & = & -1 \\ x_1 & = & 2 \\ x_2 & = & 2 \end{array}$$

solution S

- To check solution S, plug in values and verify each equation.

Search Problems

or report none exists

Search problem. Given an instance I of a problem, find a solution S.

Requirement. Must be able to efficiently check that S is a solution.

poly-time in size of instance I

LP. Given a system of linear inequalities, find a solution.

$$\begin{array}{rrrcl} 48x_0 & +16x_1 & +119x_2 & \le & 88 \\ 5x_0 & +\ 4x_1 & +\ 35x_2 & \ge & 13 \\ 15x_0 & +\ 4x_1 & +\ 20x_2 & \ge & 23 \\ x_0 & , \quad x_1 & , \quad x_2 & \ge & 0 \end{array}$$

instance I

$$\begin{array}{rcl} x_0 & = & 1 \\ x_1 & = & 1 \\ x_2 & = & 1/5 \end{array}$$

solution S

- To check solution S, plug in values and verify each inequality.

Search Problems

or report none exists

Search problem. Given an instance I of a problem, find a solution S.

Requirement. Must be able to efficiently check that S is a solution.

poly-time in size of instance I

ILP. Given a system of linear inequalities, find a binary solution.

$$\begin{array}{rcrcrcl} & & x_1 & + & x_2 & \geq & 1 \\ x_0 & & & + & x_2 & \geq & 1 \\ x_0 & + & x_1 & + & x_2 & \leq & 2 \end{array}$$

instance I

$$\begin{array}{rcl} x_0 & = & 0 \\ x_1 & = & 1 \\ x_2 & = & 1 \end{array}$$

solution S

- To check solution S, check that values are 0/1 , then plug in values and verify each inequality.

Search Problems

or report none exists

Search problem. Given an instance I of a problem, find a solution S.

Requirement. Must be able to efficiently check that S is a solution.

poly-time in size of instance I

SAT. Given a system of boolean equations, find a solution.

$$\begin{array}{lllllll} (x_0 \text{ and } x_1 \text{ and } x_2) & \text{or} & (x_1 \text{ and } x_2) & \text{or} & (x_0 \text{ and } x_2) & = & \text{true} \\ (x_0 \text{ and } x_1) & & & \text{or} & (x_1 \text{ and } x_2) & = & \text{false} \\ (x_1 \text{ and } x_2) & \text{or} & (x_0 \text{ and } x_2) & \text{or} & (x_0) & = & \text{true} \end{array}$$

instance I

$$\begin{array}{ll} x_0 & = \text{false} \\ x_1 & = \text{true} \\ x_2 & = \text{true} \end{array}$$

solution S

- To check solution S, plug in values and verify each equation.

Search Problems

or report none exists

Search problem. Given an instance I of a problem, find a solution S.

Requirement. Must be able to efficiently check that S is a solution.

poly-time in size of instance I

FACTOR. Find a nontrivial factor of the integer x.

`147573952589676412927` `193707721`

instance I solution S

- To check solution S, long divide `193707721` into `147573952589676412927`.

NP

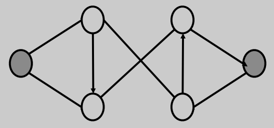
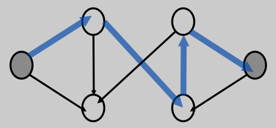
Def. NP is the class of all search problems ⟵ problems with poly-time checkable solutions

problem	description	poly-time algorithm	instance I	solution S
LSOLVE (A, b)	Find a vector x that satisfies $Ax = b$.	Gaussian elimination	$\begin{array}{rrrcr} 0x_0 & +\ 1x_1 & +\ 1x_2 & = & 4 \\ 2x_0 & +\ 4x_1 & -\ 2x_2 & = & 2 \\ 0x_0 & +\ 3x_1 & +15x_2 & = & 36 \end{array}$	$\begin{array}{rcr} x_0 & = & -1 \\ x_1 & = & 2 \\ x_2 & = & 2 \end{array}$
LP (A, b)	Find a vector x that satisfies $Ax \le b$.	ellipsoid	$\begin{array}{rrrcr} 48x_0 & +16x_1 & +119x_2 & \le & 88 \\ 5x_0 & +\ 4x_1 & +\ 35x_2 & \ge & 13 \\ 15x_0 & +\ 4x_1 & +\ 20x_2 & \ge & 23 \\ x_0 & ,\ x_1 & ,\ x_2 & \ge & 0 \end{array}$	$\begin{array}{rcr} x_0 & = & 1 \\ x_1 & = & 1 \\ x_2 & = & 1/5 \end{array}$
ILP (A, b)	Find a binary vector x that satisfies $Ax \le b$.	???	$\begin{array}{rrrcr} & x_1 & +\ x_2 & \ge & 1 \\ x_0 & & +\ x_2 & \ge & 1 \\ x_0 & +\ x_1 & +\ x_2 & \le & 2 \end{array}$	$\begin{array}{rcr} x_0 & = & 0 \\ x_1 & = & 1 \\ x_2 & = & 1 \end{array}$
SAT (A, b)	Find a boolean vector x that satisfies $Ax = b$.	???	$\begin{array}{lcl} (x_1 \textit{ and } x_2) \textit{ or } (x_0 \textit{ and } x_2) & = & \textit{true} \\ (x_0 \textit{ and } x_1) \textit{ or } (x_1 \textit{ and } x_2) & = & \textit{false} \\ (x_0 \textit{ and } x_2) \textit{ or } (x_0) & = & \textit{true} \end{array}$	$\begin{array}{lcl} x_0 & = & \textit{false} \\ x_1 & = & \textit{true} \\ x_2 & = & \textit{true} \end{array}$
FACTOR (x)	Find a nontrivial factor of the integer x.	???	8784561	10657

Significance. What scientists, engineers, and applications programmers aspire to compute feasibly.

P

Def. P is the class of search problems solvable in poly-time.
A search problem that is not in P is said to be intractable.

problem	description	poly-time algorithm	instance I	solution S
STCONN (G, s, t)	Find a path from s to t in digraph G.	depth-first search (Theseus)		
SORT (a)	Find permutation that puts a in ascending order.	mergesort (von Neumann 1945)	2.3 8.5 1.2 9.1 2.2 0.3	5 2 4 0 1 3
LSOLVE (A, b)	Find a vector x that satisfies $Ax = b$.	Gaussian elimination (Edmonds, 1967)	$\begin{array}{rrrcr} 0x_0 & + 1x_1 & + 1x_2 & = & 4 \\ 2x_0 & + 4x_1 & - 2x_2 & = & 2 \\ 0x_0 & + 3x_1 & + 15x_2 & = & 36 \end{array}$	$\begin{array}{rcr} x_0 & = & -1 \\ x_1 & = & 2 \\ x_2 & = & 2 \end{array}$
LP (A, b)	Find a vector x that satisfies $Ax \leq b$.	ellipsoid (Khachiyan, 1979)	$\begin{array}{rrrcr} 48x_0 & + 16x_1 & + 119x_2 & \leq & 88 \\ 5x_0 & + 4x_1 & + 35x_2 & \geq & 13 \\ 15x_0 & + 4x_1 & + 20x_2 & \geq & 23 \\ x_0 & , x_1 & , x_2 & \geq & 0 \end{array}$	$\begin{array}{rcr} x_0 & = & 1 \\ x_1 & = & 1 \\ x_2 & = & \frac{1}{5} \end{array}$

Significance. What scientists and engineers, and applications programmers do compute feasibly.

Other types of problems

Search problem. Find a solution.

Decision problem. Is there a solution?

Optimization problem. Find the best solution.

Some problems are more naturally formulated in one regime than another.
Ex. TSP is usually "find the shortest tour that connects all the cities."

Not technically equivalent, but main conclusions that we draw apply to all 3.

Note: Standard definitions of P and NP are in terms of decision problems.

Nondeterminism

Nondeterministic machine can guess the desired solution

Ex. `int[] a = new a[N];`

- Java: values are all 0
- nondeterministic machine: values are the answer!

ILP. Given a system of linear inequalities, guess a 0/1 solution.

$$\begin{array}{ccccccc} & & x_1 & + & x_2 & \geq & 1 \\ x_0 & & & + & x_2 & \geq & 1 \\ x_0 & + & x_1 & + & x_2 & \leq & 2 \end{array}$$

instance I

$$\begin{array}{ccc} x_0 & = & 0 \\ x_1 & = & 1 \\ x_2 & = & 1 \end{array}$$

solution S

Ex. Turing machine

- deterministic: state, input determines next state
- nondeterministic: more than one possible next state

$$\begin{tikzcd} & B \\ A \arrow[ur, "0:x"] \arrow[dr, "0:y"'] & \\ & C \end{tikzcd}$$

NP: Search problems solvable in poly time on a nondeterministic machine.

all of them!

Extended Church-Turing Thesis

Extended Church-Turing thesis.

P = search problems solvable in poly-time in this universe.

Evidence supporting thesis.
- True for all physical computers.
- Simulating one computer on another adds poly-time cost factor.
- Nondeterministic machine seems to be a fantasy.

Implication. To make future computers more efficient, suffices to focus on improving implementation of existing designs.

A new law of physics? A constraint on what is possible.
Possible counterexample? Quantum computer

P vs. NP

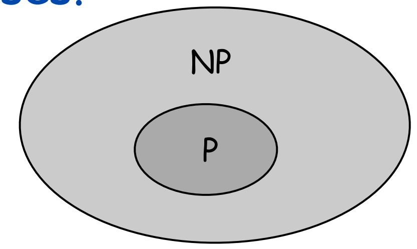
The Central Question

P. Class of search problems solvable in poly-time.
NP. Class of all search problems.

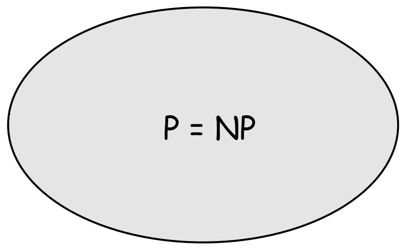
Does P = NP?

- can you always avoid brute-force search and do better??
- does nondeterminism make a computer more efficient??
- are there any intractable search problems??

Two possible universes.



P ≠ NP



P = NP

If yes… Poly-time algorithms for 3-SAT, ILP, TSP, FACTOR, …
If no… Would learn something fundamental about our universe.

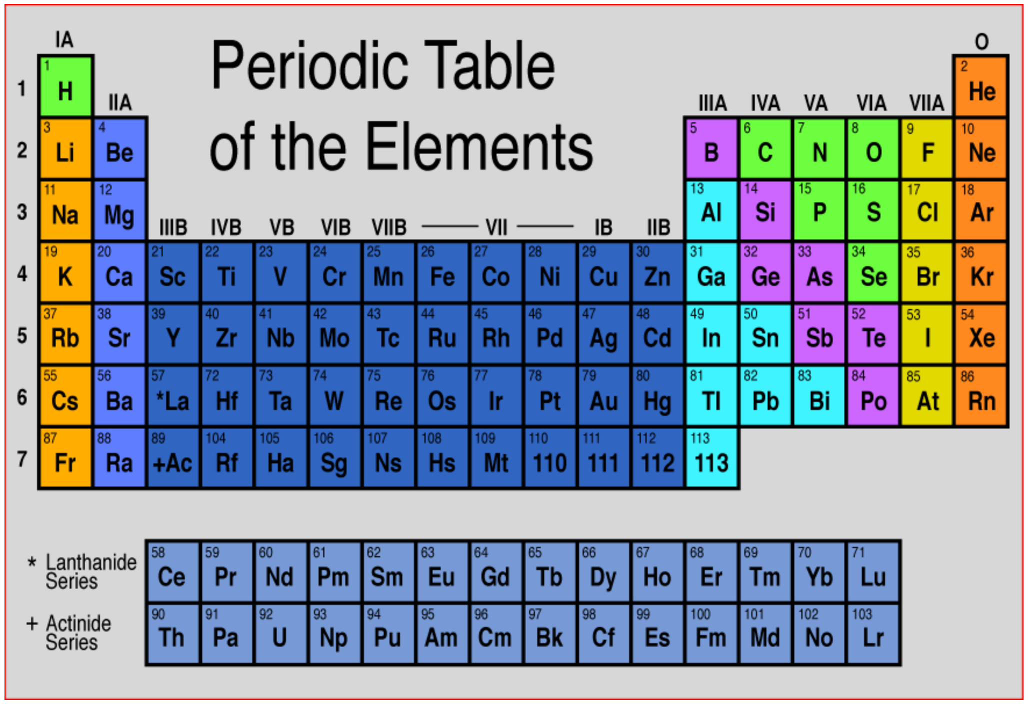
Overwhelming consensus. P ≠ NP.

Fame and Fortune through CS

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.

Classifying Problems



Periodic Table of the Elements

	IA	IIA	IIIB	IVB	VB	VIB	VIIB		VII		IB	IIB	IIIA	IVA	VA	VIA	VIIA	O
1	1 H																	2 He
2	3 Li	4 Be											5 B	6 C	7 N	8 O	9 F	10 Ne
3	11 Na	12 Mg											13 Al	14 Si	15 P	16 S	17 Cl	18 Ar
4	19 K	20 Ca	21 Sc	22 Ti	23 V	24 Cr	25 Mn	26 Fe	27 Co	28 Ni	29 Cu	30 Zn	31 Ga	32 Ge	33 As	34 Se	35 Br	36 Kr
5	37 Rb	38 Sr	39 Y	40 Zr	41 Nb	42 Mo	43 Tc	44 Ru	45 Rh	46 Pd	47 Ag	48 Cd	49 In	50 Sn	51 Sb	52 Te	53 I	54 Xe
6	55 Cs	56 Ba	57 *La	72 Hf	73 Ta	74 W	75 Re	76 Os	77 Ir	78 Pt	79 Au	80 Hg	81 Tl	82 Pb	83 Bi	84 Po	85 At	86 Rn
7	87 Fr	88 Ra	89 +Ac	104 Rf	105 Ha	106 Sg	107 Ns	108 Hs	109 Mt	110 110	111 111	112 112	113 113					

* Lanthanide Series	58 Ce	59 Pr	60 Nd	61 Pm	62 Sm	63 Eu	64 Gd	65 Tb	66 Dy	67 Ho	68 Er	69 Tm	70 Yb	71 Lu
+ Actinide Series	90 Th	91 Pa	92 U	93 Np	94 Pu	95 Am	96 Cm	97 Bk	98 Cf	99 Es	100 Fm	101 Md	102 No	103 Lr

Exhaustive Search

Q. How to solve an instance of 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 SAT.

SAT is intractable

Classifying Problems

Q. Which search problems are in P?
Q. Which search problems are not in P (intractable)?

A. No easy answers (we don't even know whether P = NP).

First step. Formalize notion:

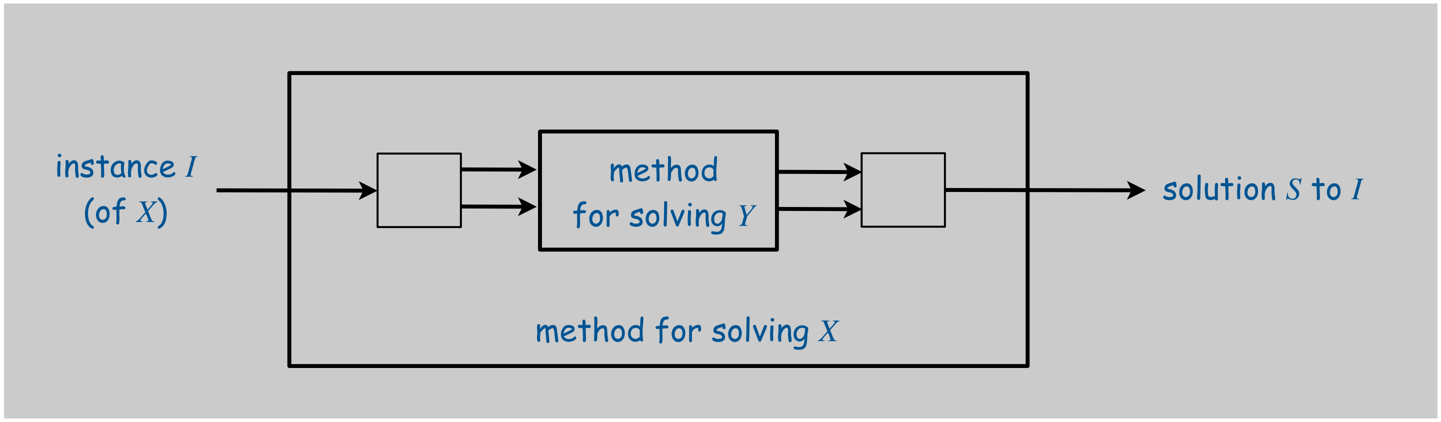
Problem X is computationally not much harder than problem Y.

Reductions

Def. Problem X reduces to problem Y if you can use an efficient solution to Y to develop an efficient solution to X

To solve X, use:

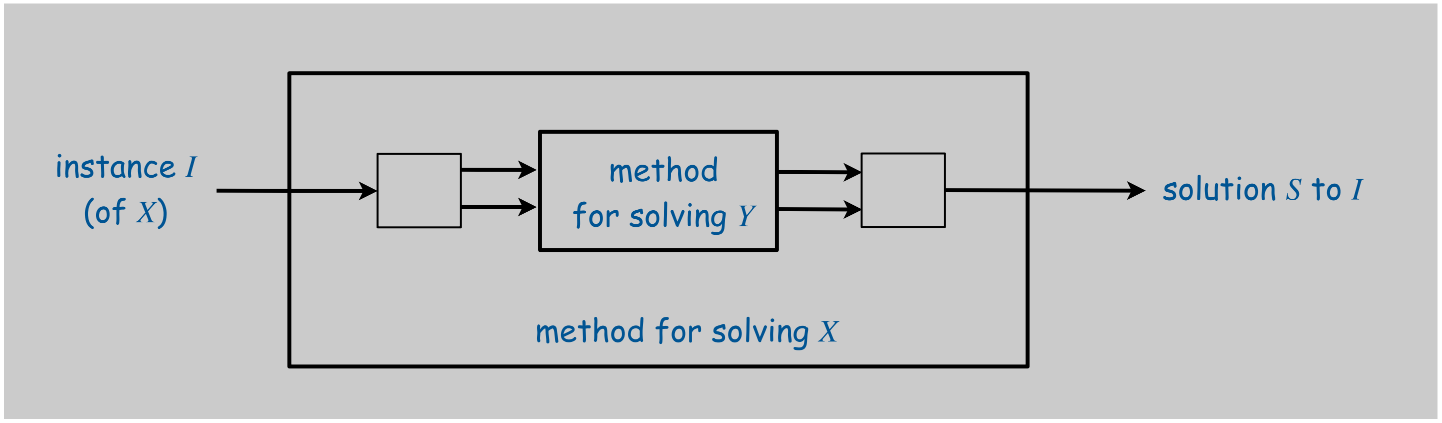
- a poly number of standard computational steps, plus
- a poly number of calls to a method that solves instances of Y.



Reductions: Consequences

Def. Problem X reduces to problem Y if you can solve X given:

- A poly number of standard computational steps, plus
- A poly number of calls to a subroutine for solving instances of Y.



previously solved problem

your research problem

Design algorithms. If poly-time algorithm for Y, then one for X too.

Establish intractability. If no poly-time algorithm for X, then none for Y.

3-SAT

your research problem

LSOLVE Reduces to LP

LSOLVE. Given a system of linear equations, find a solution.

$$\begin{array}{rcrcrcr}
0x_0 & + & 1x_1 & + & 1x_2 & = & 4 \\
2x_0 & + & 4x_1 & - & 2x_2 & = & 2 \\
0x_0 & + & 3x_1 & + & 15x_2 & = & 36
\end{array}$$

LSOLVE instance with n variables

LP. Given a system of linear inequalities, find a solution.

$$\begin{array}{rcrcrcr}
0x_0 & + & 1x_1 & + & 1x_2 & \leq & 4 \\
0x_0 & + & 1x_1 & + & 1x_2 & \geq & 4 \\
2x_0 & + & 4x_1 & - & 2x_2 & \leq & 2 \\
2x_0 & + & 4x_1 & - & 2x_2 & \geq & 2 \\
0x_0 & + & 3x_1 & + & 15x_2 & \leq & 36 \\
0x_0 & + & 3x_1 & + & 15x_2 & \geq & 36
\end{array}$$

$$\left.\begin{array}{l} \\ \end{array}\right\} \Rightarrow \quad 0x_0 + 1x_1 + 1x_1 = 4$$

corresponding LP instance with n variables and 2n inequalities

SAT Reduces to ILP

SAT. Given a boolean equation Φ, find a satisfying truth assignment.

$$\Phi = (x_1' \text{ or } x_2 \text{ or } x_3) \text{ and } (x_1 \text{ or } x_2' \text{ or } x_3) \text{ and } (x_1' \text{ or } x_2' \text{ or } x_3') \text{ and } (x_1' \text{ or } x_2' \text{ or } x_4)$$

SAT instance with n variables, k clauses

ILP. Given a system of linear inequalities, find a 0-1 solution.

$$\underbrace{\begin{aligned} C_1 &\geq 1 - x_1 \\ C_1 &\geq x_2 \\ C_1 &\geq x_3 \\ C_1 &\leq (1 - x_1) + x_2 + x_3 \end{aligned}}_{C_1 = 1 \text{ iff clause 1 is satisfied}} \qquad \underbrace{\begin{aligned} \Phi &\leq C_1 \\ \Phi &\leq C_2 \\ \Phi &\leq C_3 \\ \Phi &\leq C_4 \\ \Phi &\geq C_1 + C_2 + C_3 + C_4 - 3 \end{aligned}}_{\Phi = 1 \text{ iff } C_1 = C_2 = C_3 = C_4 = 1}$$

corresponding ILP instance with n + k + 1 variables and 4k + k + 1 inequalities
solution to this ILP instance gives solution to 3-SAT instance

More Reductions From SAT

$$\begin{tikzcd}
 & & \text{SAT} \arrow[dll, "\text{3-SAT reduces to 3-COLOR}"'] \arrow[dl] \arrow[d] \arrow[dr] & & & \\
\text{3-COLOR} \arrow[d] \arrow[dr] & \text{3DM} & \text{ILP} & \text{VERTEX COVER} \arrow[d] \arrow[dr] & & \\
\text{EXACT COVER} \arrow[d] & \text{PLANAR-3-COLOR} & & \text{CLIQUE} \arrow[d] & \text{HAM-CYCLE} \arrow[d] \arrow[dr] & \\
\text{SUBSET-SUM} \arrow[d] & & & \text{INDEPENDENT SET} & \text{TSP} & \text{HAM-PATH} \\
\text{PARTITION} \arrow[d] \arrow[dr] & & & & & \\
\text{KNAPSACK} & \text{BIN-PACKING} & & & &
\end{tikzcd}$$

Dick Karp
'85 Turing award

Conjecture: SAT is intractable.
Implication: all of these problems are intractable.

Still More Reductions from SAT

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 $a_1, \ldots, a_n$, 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, integer programming.

Physics. Partition function of 3d Ising model.

Politics. Shapley-Shubik voting power.

Pop culture. Versions of Sudoko, Checkers, Minesweeper, Tetris.

Statistics. Optimal experimental design.

Conjecture: no poly-time algorithm for SAT.
Implication: all of these problems are intractable.

6,000+ scientific papers per year.

NP-completeness

NP-Completeness

Q. Why do we believe SAT has no poly-time algorithm?

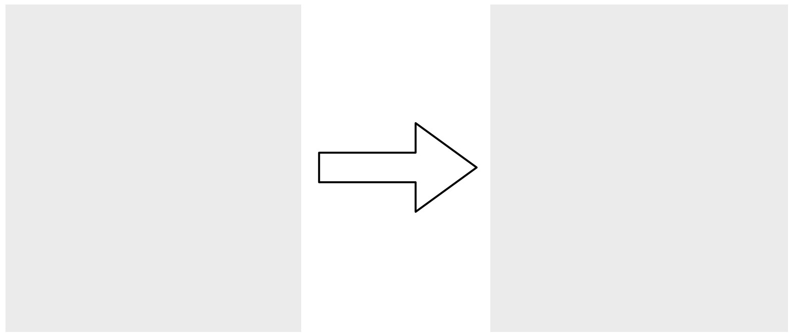
Def. An NP problem is NP-complete if all problems in NP reduce to it.

every NP problem is a 3-SAT problem in disguise

Theorem. [Cook 1971] SAT is NP-complete.

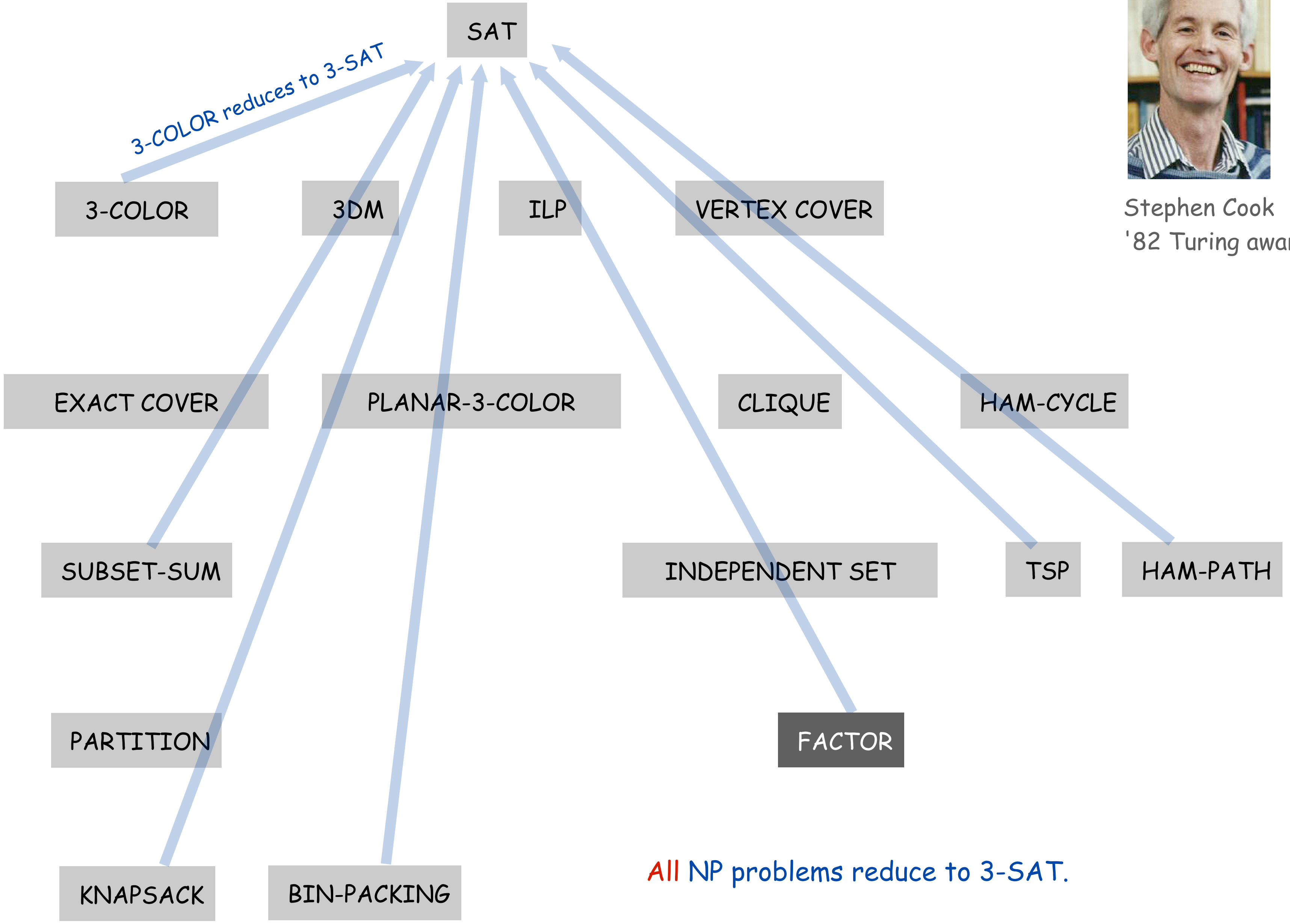
Extremely brief Proof Sketch:

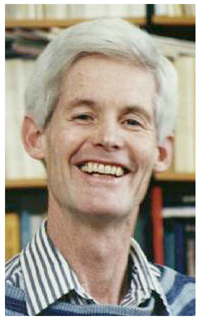
- convert non-deterministic TM notation to SAT notation
- if you can solve 3-SAT, you can solve any problem in NP



Corollary. Poly-time algorithm for SAT $\Rightarrow$ P = NP.

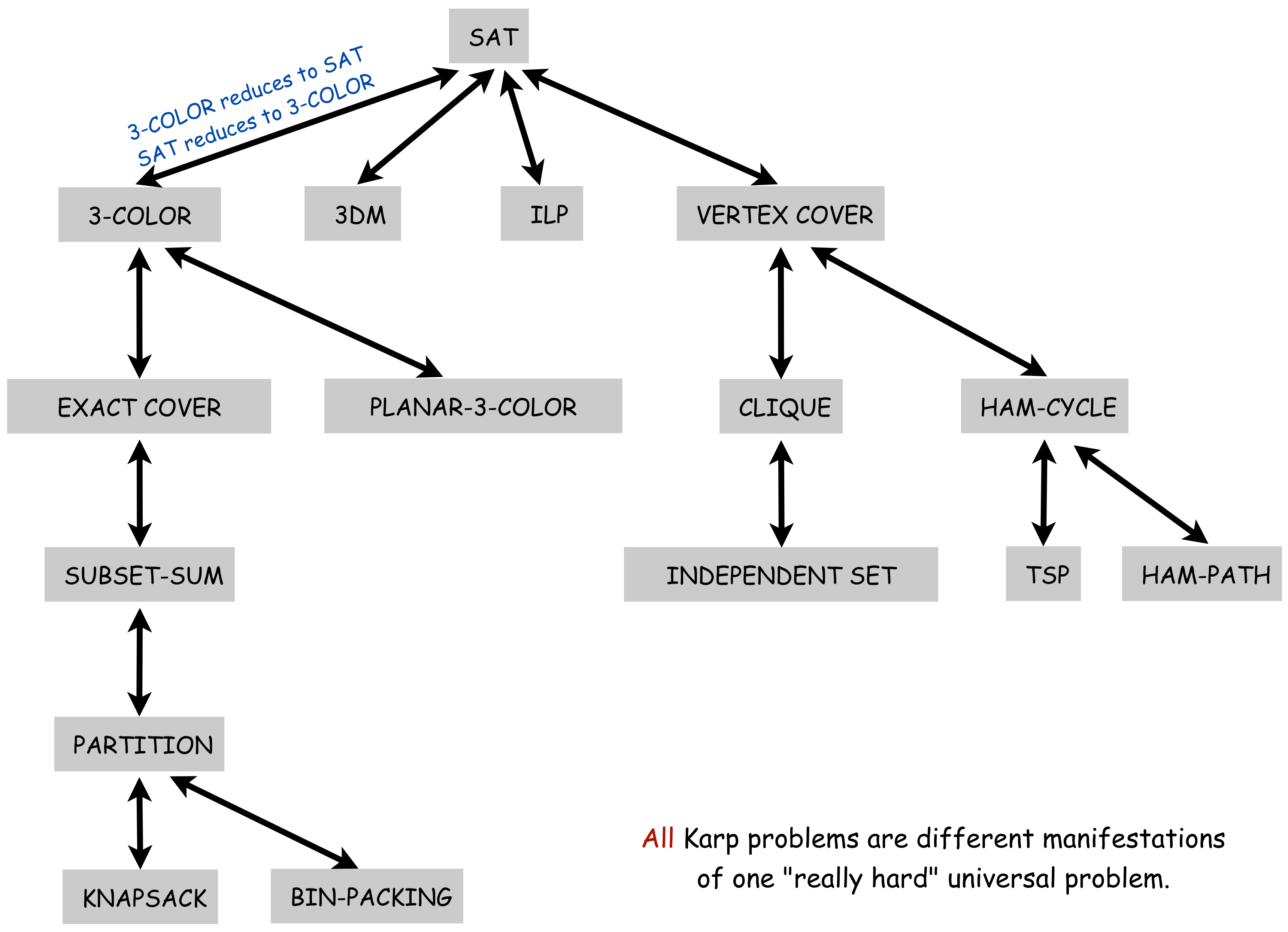
Cook's Theorem




Stephen Cook
'82 Turing award

All NP problems reduce to 3-SAT.

Cook + Karp



All Karp problems are different manifestations of one "really hard" universal problem.

Two possible universes

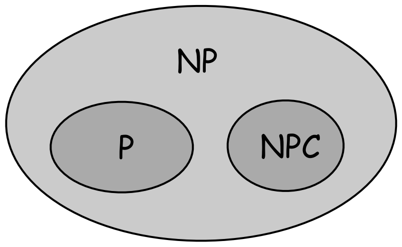
$P \neq NP$.

- Intractable search problems exist.
- Nondeterminism makes machines more efficient.
- Can prove that a problem is intractable by reduction from an NP-complete problem.
- Some search problems are neither NP-complete or in P.
- Some search problems are still not classified.

no other way is known!

we don't know any useful ones

examples: factoring, graph isomorphism

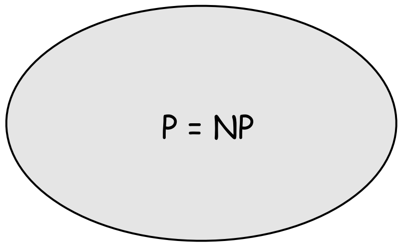


$P \neq NP$

$P = NP$.

- No intractable search problems exist.
- Nondeterminism is no help.
- Poly-time solutions exist for NP-complete problems

and all other search problems, such as factoring and graph isomorphism



$P = NP$

[Third possibility: Extended Church-Turing thesis is wrong.]

Implications of NP-completeness

Implication. [SAT captures difficulty of whole class NP.]

- Poly-time algorithm for SAT iff P = NP (no intractable search problems exist).
- If some search problem is intractable, then so is SAT.

Remark. Can replace SAT above with any NP-complete problem.

Example: Proving a problem NP-complete guides scientific inquiry.

- 1926: Ising introduces simple model for phase transitions.
- 1944: Onsager finds closed form solution to 2D version in tour de force.
- 19xx: Feynman and other top minds seek 3D solution.
- 2000: SAT reduces to 3D-ISING.

↖ a holy grail of statistical mechanics

↖ search for closed formula appears doomed since 3D-ISING is intractable if $P \neq NP$

Coping With Intractability

Coping With Intractability

You have an NP-complete problem.

- It's safe to assume that it is intractable.
- What to do?

Relax one of desired features.

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

Complexity theory deals with worst case behavior.

- Instance(s) you want to solve may have easy-to-find answer.
- `Chaff` solves real-world SAT instances with ~ 10k variables.
  [Matthew Moskewicz '00, Conor Madigan '00, Sharad Malik]

PU senior independent work (!)

Coping With Intractability

You have an NP-complete problem.

- It's safe to assume that it is intractable.
- What to do?

Relax one of desired features.

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

Develop a heuristic, and hope it produces a good solution.

- No guarantees on quality of solution.
- Ex. TSP assignment heuristics.
- Ex. Metropolis algorithm, simulating annealing, genetic algorithms.

Approximation algorithm. Find solution of provably good quality.

- Ex. MAX-3SAT: provably satisfy 87.5% as many clauses as possible.

but if you can guarantee to satisfy 87.51% as many clauses as possible in poly-time, then P = NP !

Coping With Intractability

You have an NP-complete problem.

- It's safe to assume that it is intractable.
- What to do?

Relax one of desired features.

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

Special cases may be tractable.

- Ex: Linear time algorithm for 2-SAT.
- Ex: Linear time algorithm for Horn-SAT.

each clause has at most one un-negated literal

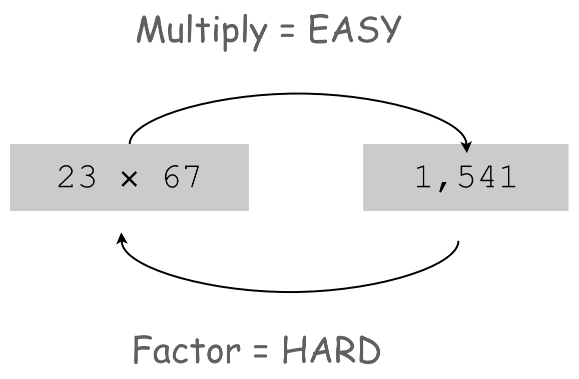
Exploiting Intractability: Cryptography

Modern cryptography.

- Ex. Send your credit card to Amazon.
- Ex. Digitally sign an e-document.
- Enables freedom of privacy, speech, press, political association.

RSA cryptosystem.

- To use: multiply two n-bit integers. [poly-time]
- To break: factor a $2n$-bit integer. [unlikely poly-time]



Exploiting Intractability: Cryptography

FACTOR. Given an n-bit integer x, find a nontrivial factor.

not 1 or x

```
740375634795617128280467960974295731425931888892312890849362
326389727650340282662768919964196251178439958943305021275853
701189680982867331732731089309005525051168770632990723963807
86710086096962537934650563796359
```

Q. What is complexity of FACTOR?

A. In NP, but not known (or believed) to be in P or NP-complete.

Q. Is it safe to assume that FACTOR is intractable?

A. Maybe, but not as safe an assumption as for an NP-complete problem.

Summary

P. Class of search problems solvable in poly-time.
NP. Class of all search problems, some of which seem wickedly hard.
NP-complete. Hardest problems in NP.
Intractable. Search problems not in P (if P $\neq$ NP).

Many fundamental problems are NP-complete

- TSP, SAT, 3-COLOR, ILP, (and thousands of others)
- 3D-ISING.

Use theory as a guide.

- An efficient algorithm for an NP-complete problem would be a stunning scientific breakthrough (a proof that P = NP)
- You will confront NP-complete problems in your career.
- It is safe to assume that P $\neq$ NP and that such problems are intractable.
- Identify these situations and proceed accordingly.