7.8 Intractability

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.
- Useful in practice ("efficient") = polynomial time for all inputs.

↖ $a N^b$

> Ex 1. Sorting N elements takes N^2 steps using insertion sort.
>
> Ex 2. Finding best TSP tour on N elements takes $N!$ steps using exhaustive search.

Theory. Definition is broad and robust.
Practice. Poly-time algorithms scale to huge problems.

↖ constants a and b tend to be small

Exponential Growth

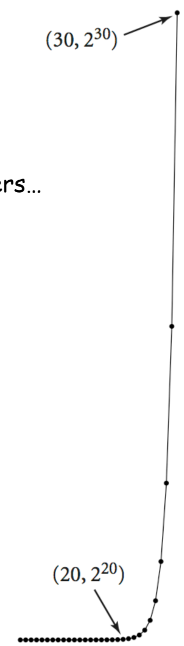
Exponential growth dwarfs technological change.
- Suppose you have a giant parallel computing device…
- With as many processors as electrons in the universe…
- And each processor has power of today's supercomputers…
- And each processor works for the life of the universe…

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 brute force.

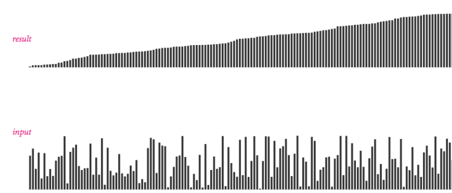
↖ $1000! \gg 10^{1000} \gg 10^{79} \times 10^{13} \times 10^{17}$



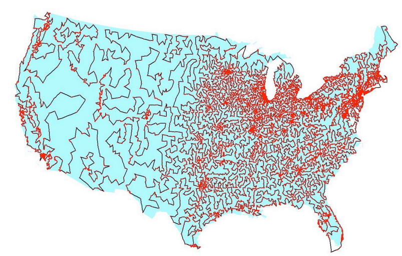
Reasonable Questions about Problems

Q. Which problems can we solve in practice?
A. Those with 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

Three Fundamental Problems

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

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

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

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

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

each x_i is either 0 or 1

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

Three 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.

Q. Which of these problems have 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. ← open problem for decades
? ILP. No poly-time algorithm known or believed to exist!

Search Problems

Search problem. Given an instance I of a problem, find a solution S. (or report none exists)
Requirement. Must be able to efficiently check that S is a solution. (poly-time in size of instance I)

Search Problems

Search problem. Given an instance I of a problem, find a solution S. (or report none exists)
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.

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

instance I — 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.

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

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.

$$\begin{array}{rcrcrcl} & & x_1 & + & x_2 & \ge & 1 \\ x_0 & & & + & x_2 & \ge & 1 \\ x_0 & + & x_1 & + & x_2 & \le & 2 \end{array}$$

instance I

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

solution S

- To check solution S, plug in values and verify each inequality (and check that solution is 0/1).

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.

input size = number of bits

`147573952589676412927`

instance I

`193707721`

solution S

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

NP

Def. NP is the class of all search problems.

classic definition limits NP to yes-no problems

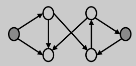
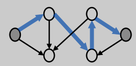
problem	description	poly-time algorithm	instance I	solution S
LSOLVE (A, b)	Find a vector x that satisfies $Ax = b$.	Gaussian elimination	$0x_0 + 1x_1 + 1x_2 = 4$ $2x_0 + 4x_1 - 2x_2 = 2$ $0x_0 + 3x_1 + 15x_2 = 36$	$x_0 = -1$ $x_1 = 2$ $x_2 = 2$
LP (A, b)	Find a vector x that satisfies $Ax \le b$.	ellipsoid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

Significance. What scientists and engineers aspire to compute feasibly.

P

Def. P is the class of search problems solvable in poly-time.

classic definition limits P to yes-no problems

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)	$0x_0 + 1x_1 + 1x_2 = 4$ $2x_0 + 4x_1 - 2x_2 = 2$ $0x_0 + 3x_1 + 15x_2 = 36$	$x_0 = -1$ $x_1 = 2$ $x_2 = 2$
LP (A, b)	Find a vector x that satisfies $Ax \le b$.	ellipsoid (Khachiyan, 1979)	$48x_0 + 16x_1 + 119x_2 \le 88$ $5x_0 + 4x_1 + 35x_2 \ge 13$ $15x_0 + 4x_1 + 20x_2 \ge 23$ $x_0, x_1, x_2 \ge 0$	$x_0 = 1$ $x_1 = 1$ $x_2 = \frac{1}{5}$

Significance. What scientists and engineers compute feasibly.

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.

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 computers.

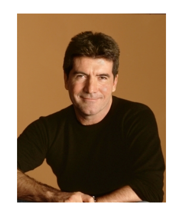
P vs. NP

Automating Creativity

Q. Being creative vs. appreciating creativity?

Ex. Mozart composes a piece of music; our neurons appreciate it.
Ex. Wiles proves a deep theorem; a colleague referees it.
Ex. Boeing designs an efficient airfoil; a simulator verifies it.
Ex. Einstein proposes a theory; an experimentalist validates it.

creative



ordinary

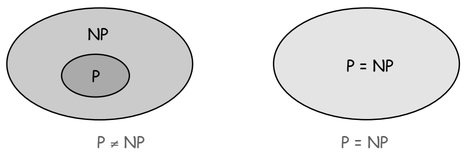
Computational analog. Does P = NP?

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 searching and do better?*

Two worlds.

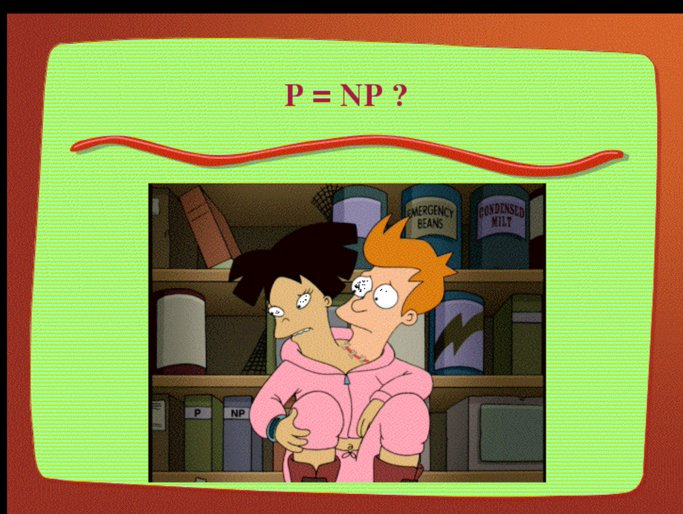


If yes... Poly-time algorithms for 3-SAT, ILP, TSP, FACTOR, ...
If no... Would learn something fundamental about our universe.

Overwhelming consensus. P ≠ NP.

The Simpsons: P = NP?

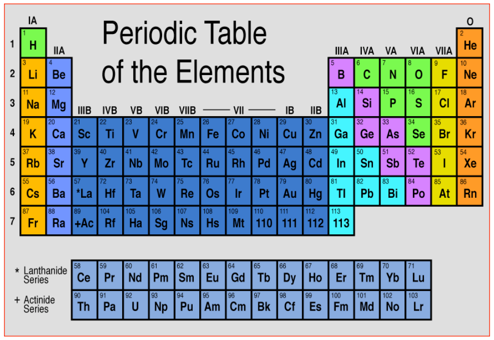
Futurama: P = NP?



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



A Hard Problem: 3-Satisfiability

Literal. A Boolean variable or its negation. $x_i\ ,\ \ x_i'$

Clause. An *or* of 3 distinct literals. $C_j\ = x_1 \text{ or } x_2' \text{ or } x_3$

Conjunctive normal form. An *and* of clauses. $\Phi\ =\ C_1 \text{ and } C_2 \text{ and } C_3 \text{ and } C_4$

3-SAT. Given a CNF formula Φ consisting of k clauses over N variables, find a satisfying truth assignment (if one exists).

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

yes: $x_1 = \text{true},\ x_2 = \text{true},\ x_3 = \text{false},\ x_4 = \text{true}$

Key application. Electronic design automation (EDA).

Exhaustive Search

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.

"intractable"

Classifying Problems

Q. Which search problems are in P?
A. No easy answers (we don't even know whether P = NP).

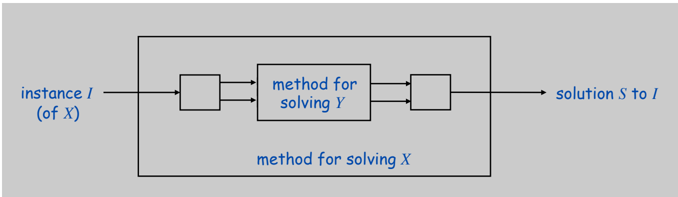
Goal. Formalize notion:

Problem X is computationally not much harder than problem Y.

Reductions

"Cook reduction"

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

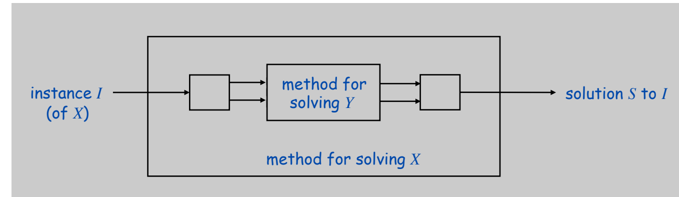


To solve X, use:

- 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 use an efficient solution to Y to develop an efficient solution to X:



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.

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

LSOLVE instance with n variables

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

$$
\begin{aligned}
0x_0 + 1x_1 + 1x_2 &\le 4\\
0x_0 + 1x_1 + 1x_2 &\ge 4\\
2x_0 + 4x_1 - 2x_2 &\le 2\\
2x_0 + 4x_1 - 2x_2 &\ge 2\\
0x_0 + 3x_1 + 15x_2 &\le 36\\
0x_0 + 3x_1 + 15x_2 &\ge 36
\end{aligned}
$$

$\} \Rightarrow 0x_0 + 1x_1 + 1x_1 = 4$

corresponding LP instance with n variables and 2n inequalities

3-SAT Reduces to ILP

3-SAT. Given a CNF formula Φ, find a satisfying truth assignment.

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

3-SAT instance with n variables, k clauses

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

$$
\begin{aligned}
C_1 &\ge 1 - x_1\\
C_1 &\ge x_2\\
C_1 &\ge x_3\\
C_1 &\le (1 - x_1) + x_2 + x_3
\end{aligned}
$$

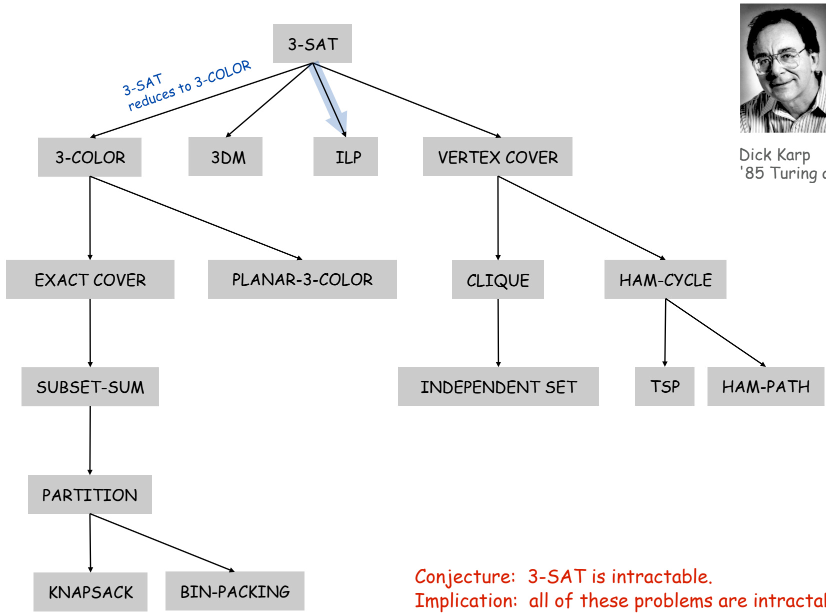
$C_1 = 1$ iff clause 1 is satisfied (similar inequalities for C_2, C_3, and C_4)

$$
\begin{aligned}
\Phi &\le C_1\\
\Phi &\le C_2\\
\Phi &\le C_3\\
\Phi &\le C_4\\
\Phi &\ge C_1 + C_2 + C_3 + C_4 - 3
\end{aligned}
$$

$\Phi = 1$ 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 original 3-SAT instance)

More Reductions From 3-SAT



Dick Karp
'85 Turing award

Conjecture: 3-SAT is intractable.
Implication: all of these problems are intractable.

Still More Reductions from 3-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.

6,000+ scientific papers per year.

Conjecture: 3-SAT is intractable.
Implication: all of these problems are intractable.

NP-completeness

NP-Completeness

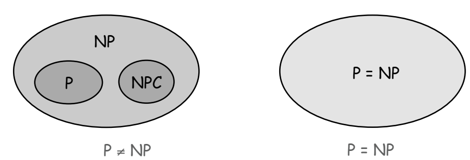
Q. Why do we believe 3-SAT is intractable?

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

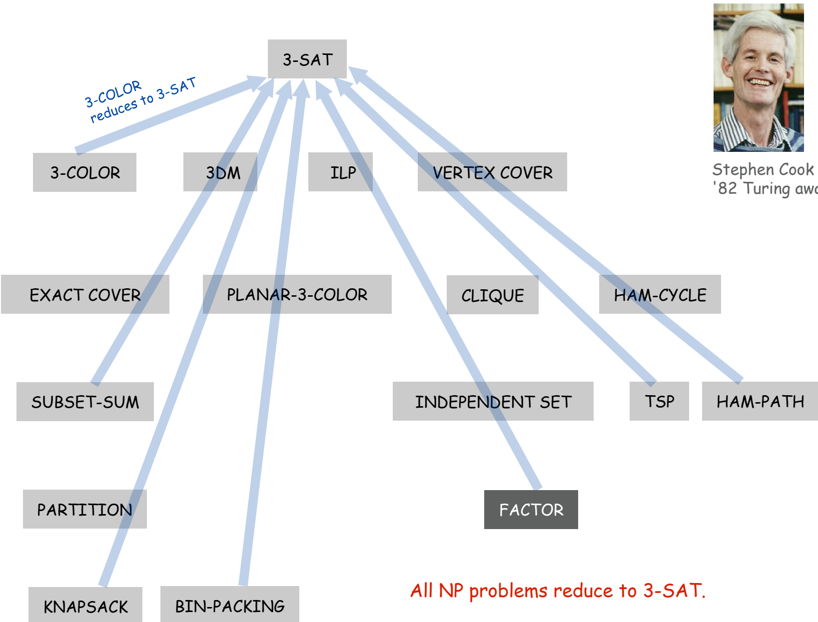
every NP problem is a 3-SAT problem in disguise

Theorem. [Cook 1971] 3-SAT is NP-complete.
Corollary. Poly-time algorithm for 3-SAT $\Rightarrow$ P = NP.

Two worlds.



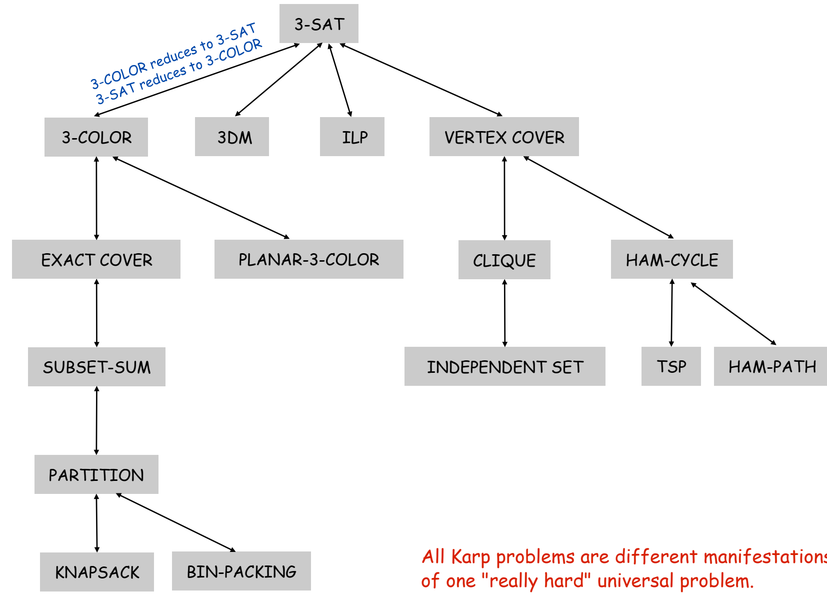
Cook's Theorem



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.

Implications of NP-Completeness

Implication. [3-SAT captures difficulty of whole class NP.]

- Poly-time algorithm for 3-SAT iff P = NP.
- If no poly-time algorithm for some NP problem, then none for 3-SAT.

Remark. Can replace 3-SAT with any of Karp's problems.

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: 3D-ISING is NP-complete.

a holy grail of statistical mechanics

search for closed formula appears doomed

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. Problem with no poly-time algorithm.

Many fundamental problems are NP-complete.

- TSP, 3-SAT, 3-COLOR, ILP.
- 3D-ISING.

Use theory a guide:

- A poly-time algorithm for an NP-complete problem would be a stunning breakthrough (a proof that P = NP).
- You will confront NP-complete problems in your career.
- Safe to assume that P ≠ NP and that such problems are intractable.
- Identify these situations and proceed accordingly.

Coping With NP-completeness

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 be "easy."
- `Chaff` solves real-world SAT instances with ~ 10k variable.
  [Matthew Moskewicz '00, Conor Madigan '00, Sharad Malik]

PU senior independent work (!)

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 !

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

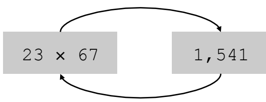
Modern cryptography.

- 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]

Multiply = EASY



Factor = HARD

Exploiting Intractability: Cryptography

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

not 1 or x

```
74037563479561712828046796097429573142593188889231289084936232638972765034028266276891996419625117843995894330502127585370118968098286733173273108930900552505116877063299072396380786710086096962537934650563796359
```

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

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 FACTOR is intractable?
A. Maybe, but not as safe as assumption for NP-complete.

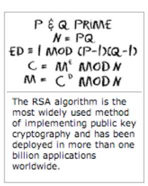
Fame and Fortune through CS (revisited)

Challenge. Factor this number.

```
74037563479561712828046796097429573142593188889231289084936232638972765034028266276891996419625117843995894330502127585370118968098286733173273108930900552505116877063299072396380786710086096962537934650563796359
```

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

Can't do it? Create a company based on the difficulty of factoring.



RSA algorithm

RSA sold for $2.1 billion

or design a t-shirt

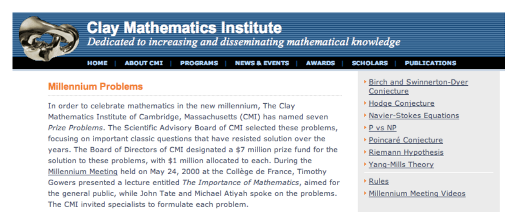
Fame and Fortune through CS (revisited)

Challenge. Factor this number.

```
74037563479561712828046796097429573142593188889231289084936232638972765034028266276891996419625117843995894330502127585370118968098286733173273108930900552505116877063299072396380786710086096962537934650563796359
```

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

Can't do it? Try resolving P = NP question (need more math and cs).



$1 million prize

A Final Thought

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

not 1 or x

Q. What is complexity of FACTOR?
A. In NP, but not known (or believed) to be in P or NP-complete.

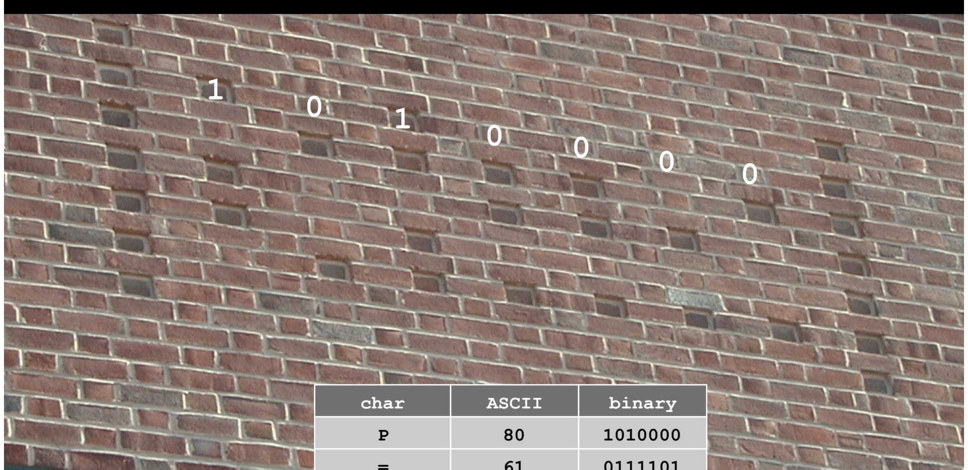
Q. What if P = NP?
A. Poly-time algorithm for factoring; modern e-conomy collapses.

Quantum. [Shor 1994] Can factor an n-bit integer in n^3 steps on a "quantum computer."

Q. Do we still believe the extended Church-Turing thesis???

CS Building, West Wall

Princeton CS Building, West Wall, Circa 2001



char	ASCII	binary
P	80	1010000
=	61	0111101
N	78	1001110
P	80	1010000
?	63	0111111