

## 7.8 Intractability



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.

$an^b$

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 large problems.

constants  $a$  and  $b$  tend to be small

2

## 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 <sup>†</sup>	$10^{79}$
supercomputer instructions per second	$10^{13}$
age of universe in seconds <sup>†</sup>	$10^{17}$

<sup>†</sup> estimated

- Will not help solve 1,000 city TSP problem via brute force.

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



3

## 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.

4

## Three Fundamental Problems

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

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

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

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

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

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

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

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

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

each  $x_i$  is either 0 or 1

5

## 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 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.** No poly-time algorithm known or believed to exist!

6

## Search Problems

**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  $I$

or report none exists



7

## Search Problems

**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  $I$

or report none exists

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

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

instance  $I$

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

solution  $S$

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

8

## Search Problems

**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.

or report none exists  
 poly-time in size of instance  $I$

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

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

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.

9

## Search Problems

**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.

or report none exists  
 poly-time in size of instance  $I$

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

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

instance  $I$

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

solution  $S$

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

10

## Search Problems

**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.

or report none exists  
 poly-time in size of instance  $I$

**FACTOR.** Find a nontrivial factor of the integer  $x$ .

147573952589676412927

instance  $I$

193707721

solution  $S$

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

11

## NP

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

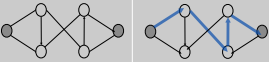
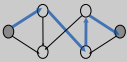
← slightly non-standard definition

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

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

12

**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**. ← slightly non-standard definitions

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 $Ax \leq b$ .	ellipsoid (Khachiyan, 1979)	$48x_0 + 16x_1 + 119x_2 \leq 88$ $5x_0 + 4x_1 + 35x_2 \geq 13$ $15x_0 + 4x_1 + 20x_2 \geq 23$ $x_0, x_1, x_2 \geq 0$	$x_0 = 1$ $x_1 = 1$ $x_2 = \frac{1}{5}$

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

13

## 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

15

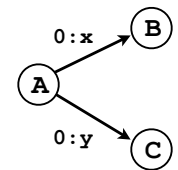
Nondeterministic machine can **guess** the desired solution

**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 binary solution.

$x_1 + x_2 \geq 1$	$x_0 = 0$
$x_0 + x_2 \geq 1$	$x_1 = 1$
$x_0 + x_1 + x_2 \leq 2$	$x_2 = 1$
instance $I$	solution $S$



**Ex. Turing machine**

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

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

← all of them!

14

## P vs. NP

16

## 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  $P = NP$ ?

17

## The Central Question

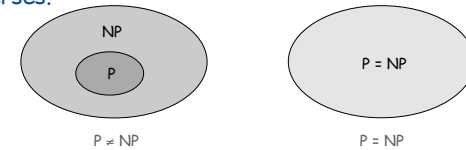
P. Class of search problems solvable in poly-time.

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.



If yes... Poly-time algorithms for 3-SAT, ILP, TSP, FACTOR, ...

If no... Would learn something fundamental about our universe.

Overwhelming consensus.  $P \neq NP$ .

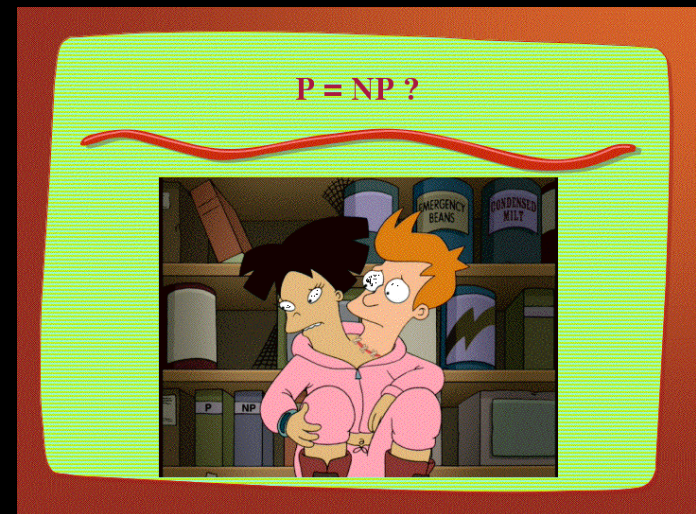
18

### $P = NP?$ in Popular Culture: The Simpsons



Copyright © 1990, Matt Groening

### $P = NP?$ in Popular Culture: Futurama



Copyright © 2000, Twentieth Century Fox

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

The periodic table is organized into groups (IA to VIIA) and periods (1 to 7). It includes the Lanthanide Series and Actinide Series at the bottom.

21

22

### 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  $\Phi$  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.

3-SAT is "intractable"



23

24



## 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.*

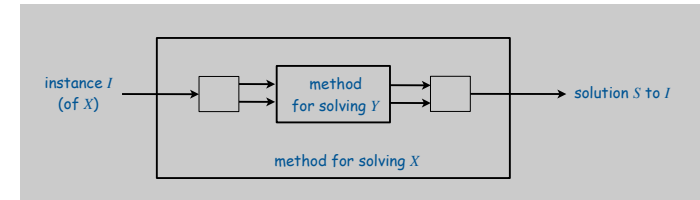
25

## 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:

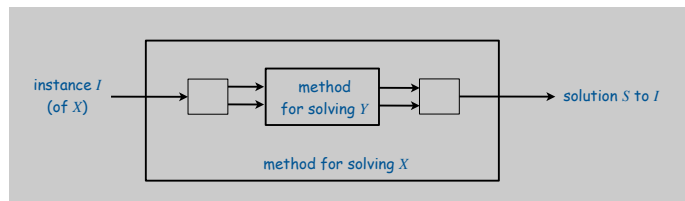
- a poly number of standard computational steps, plus
- a poly number of calls to a method that solves instances of  $Y$ .



26

## 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  $Y$ .



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

previously solved problem      your research problem

27

## LSOLVE Reduces to LP

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

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

LSOLVE instance with  $n$  variables

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

$$\left. \begin{array}{rcl} 0x_0 & + & 1x_1 + 1x_2 \leq 4 \\ 0x_0 & + & 1x_1 + 1x_2 \geq 4 \\ 2x_0 & + & 4x_1 - 2x_2 \leq 2 \\ 2x_0 & + & 4x_1 - 2x_2 \geq 2 \\ 0x_0 & + & 3x_1 + 15x_2 \leq 36 \\ 0x_0 & + & 3x_1 + 15x_2 \geq 36 \end{array} \right\} \Rightarrow 0x_0 + 1x_1 + 1x_2 = 4$$

corresponding LP instance with  $n$  variables and  $2n$  inequalities

28

## 3-SAT Reduces to ILP

**3-SAT.** Given a CNF formula  $\Phi$ , 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

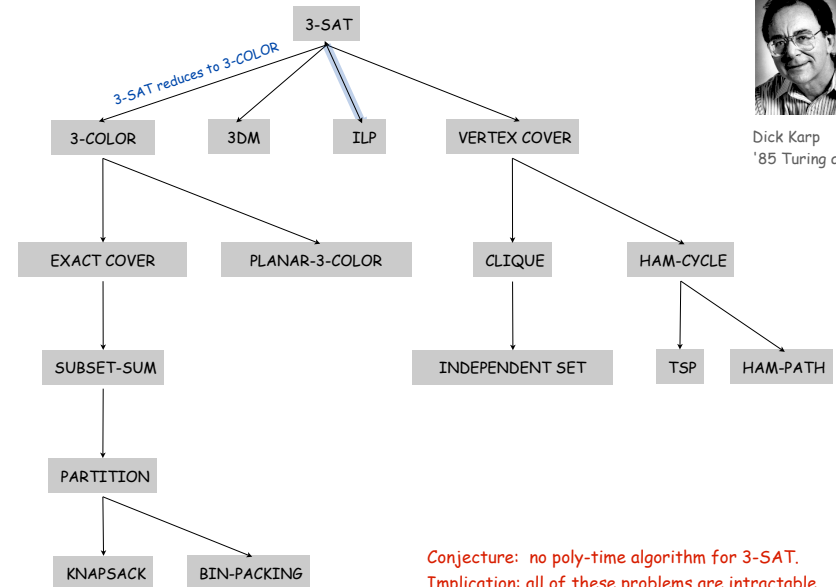
**ILP.** Given a system of linear inequalities, find a binary solution.

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

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

29

## More Reductions From 3-SAT



Dick Karp  
'85 Turing award

30

## 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  $a_1, \dots, a_n$ , compute  $\int_0^{2\pi} \cos(a_1\theta) \times \cos(a_2\theta) \times \dots \times \cos(a_n\theta) d\theta$

**Mechanical engineering.** Structure of turbulence in sheared flows.

**Medicine.** Reconstructing 3d shape from biplane angiogram.

**Operations research.** Traveling salesperson problem, integer programming.

**Physics.** Partition function of 3d Ising model.

**Politics.** Shapley-Shubik voting power.

**Pop culture.** Versions of Sudoku, Checkers, Minesweeper, Tetris.

**Statistics.** Optimal experimental design.

Conjecture: no poly-time algorithm for 3-SAT.  
Implication: all of these problems are intractable.

6,000+ scientific papers per year.

31

## NP-completeness

32



## NP-Completeness

Q. Why do we believe 3-SAT has no poly-time algorithm?

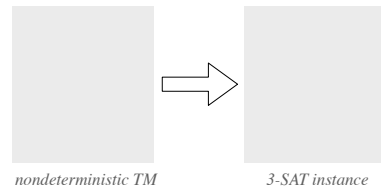
Def. An NP problem is **NP-complete** if all problems in NP reduce to it.

every NP problem is a 3-SAT problem in disguise

Theorem. [Cook 1971] 3-SAT is NP-complete.

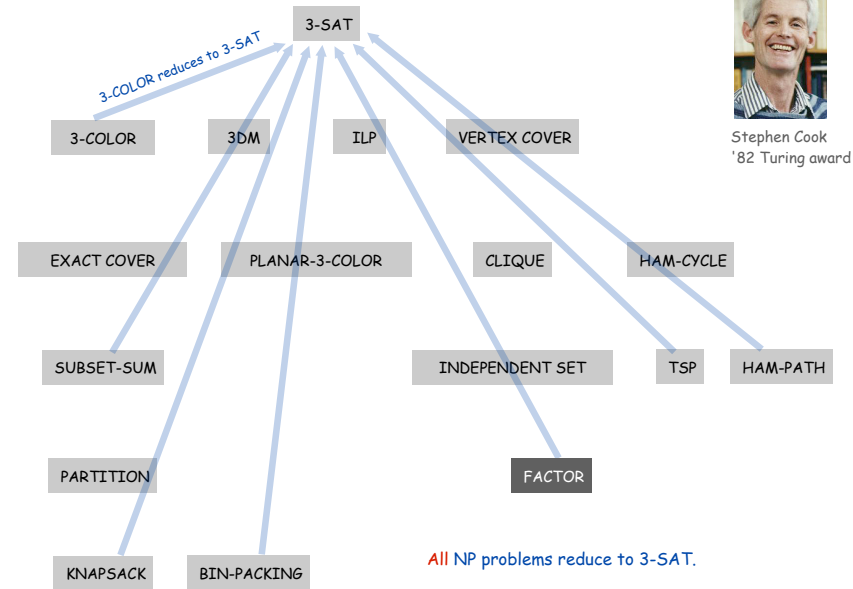
Extremely brief Proof Sketch:

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



Corollary. Poly-time algorithm for 3-SAT  $\Rightarrow$  P = NP.

## Cook's Theorem

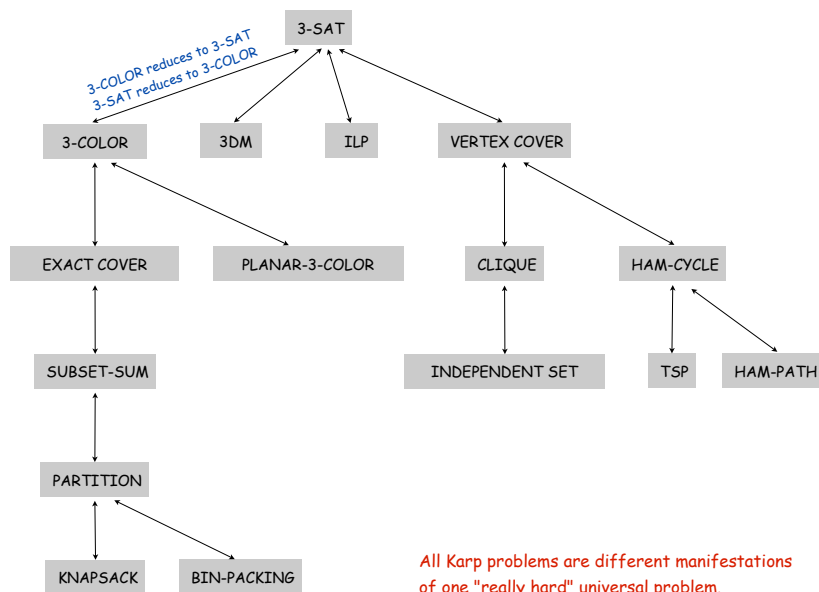


Stephen Cook  
'82 Turing award

33

34

## Cook + Karp

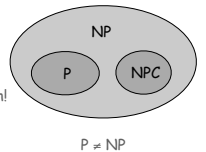


35

## Two possible universes

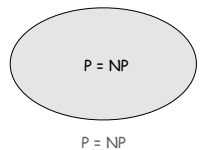
$P \neq NP$ .

- Intractable search problems exist.
- Nondeterminism makes machines more efficient.
- Can prove that a problem is intractable by reduction from an NP-complete problem. no other way is known!
- Some search problems are neither NP-complete or in P. we don't know any useful ones
- Some search problems are still not classified. examples: factoring, graph isomorphism



$P = NP$ .

- 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



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

36

## Implications of NP-completeness

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

- Poly-time algorithm for 3-SAT iff  $P = NP$  (no intractable search problems exist).
- If some search problem is intractable, then so is 3-SAT.

**Remark.** Can replace 3-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: 3-SAT reduces to 3D-ISING.

↖ a holy grail of statistical mechanics

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

37

## 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  $P \neq NP$ ).

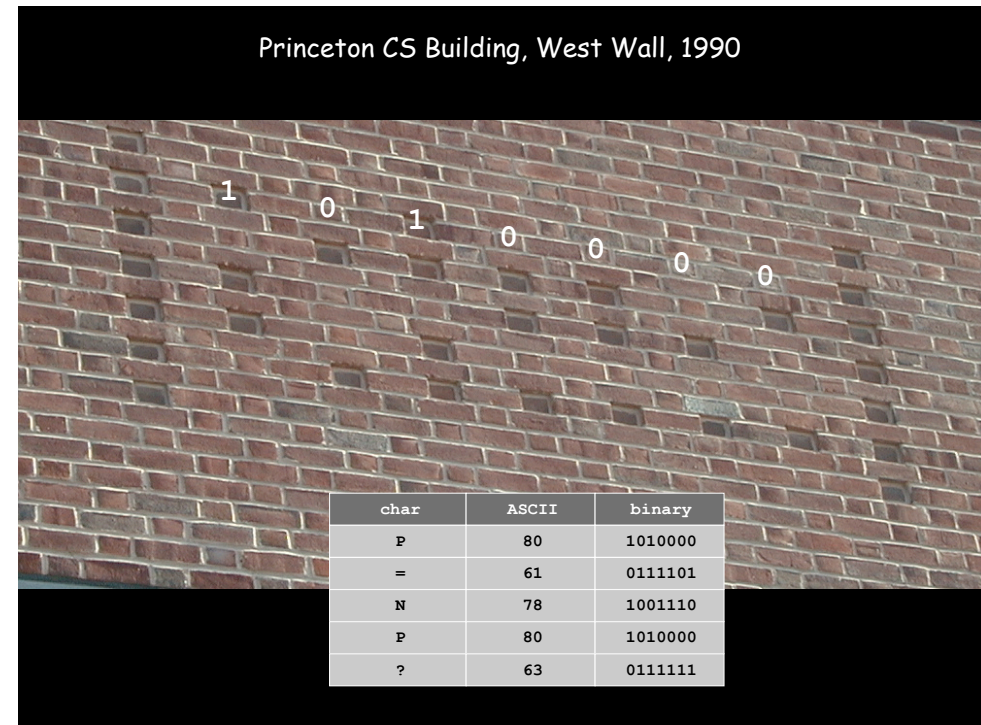
**Many fundamental problems are NP-complete**

- TSP, 3-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.

38



# 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 (!)

41

42

## 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  $P = NP$  !

43

## 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

44

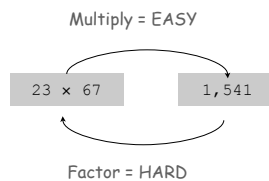
## Exploiting Intractability: Cryptography

### 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]



45

## Fame and Fortune through CS (revisited)

### Factor this number:

740375634795617128280467960974295731425931888892312890849362  
326389727650340282662768919964196251178439958943305021275853  
701189680982867331732731089309005525051168770632990723963807  
86710086096962537934650563796359

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

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

#### RSA algorithm

$P \& Q \text{ PRIME}$   
 $N = PQ$   
 $ED = I \text{ MOD } (P-1)(Q-1)$   
 $C = M^E \text{ MOD } N$   
 $M = C^D \text{ MOD } N$

The RSA algorithm is the most widely used method of implementing public key cryptography and has been deployed in more than one billion applications worldwide.



RSA sold to EMC for  
\$2.1 billion



or, sell T-shirts

47

## 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.

46

## Fame and Fortune through CS (revisited)

### Factor this number:

740375634795617128280467960974295731425931888892312890849362  
326389727650340282662768919964196251178439958943305021275853  
701189680982867331732731089309005525051168770632990723963807  
86710086096962537934650563796359

Too late? Try resolving  $P = NP?$  question (might need a few math courses).

**Clay Mathematics Institute**  
Dedicated to increasing and disseminating mathematical knowledge

HOME | ABOUT CMI | PROGRAMS | NEWS & EVENTS | AWARDS | SCHOLARS | PUBLICATIONS

**Millennium Problems**

In order to celebrate mathematics in the new millennium, The Clay Mathematics Institute of Cambridge, Massachusetts (CMI) has named seven *Prize Problems*. The Scientific Advisory Board of CMI selected these problems, focusing on important classic questions that have resisted solution over the years. The Board of Directors of CMI designated a \$7 million prize fund for the solution to these problems, with \$1 million allocated to each. During the *Millennium Meeting* held on May 24, 2000 at the Collège de France, Timothy Gowers presented a lecture entitled *The Importance of Mathematics*, aimed for the general public, while John Tate and Michael Atiyah spoke on the problems. The CMI invited specialists to formulate each problem.

- Birch and Swinnerton-Dyer Conjecture
- Hodge Conjecture
- Navier-Stokes Equations
- P vs NP
- Poincaré Conjecture
- Riemann Hypothesis
- Yang-Mills Theory
- Rules
- Millennium Meeting Videos

Clay Institute (\$1 million prize)

plus untold riches for breaking e-commerce

48

## A Final Thought

**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.** 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."

Do we still believe the extended Church-Turing thesis?