Announcements

We changed submission time for the final project to **11:59pm tonight**.
When you see the TAs, please say **thank you**! They fielded more than 100 Piazza posts yesterday alone, and they advocated for the extension.

Exam Review Session on Sunday December 13. We will release a practice exam.

Final exam is on Tuesday December 15 at **9am**. 3 locations:

- Skirkanich Auditorium: Last names A-G
- Levine Hall 101 (Wu and Chen): Last names H-P
- LRSM Auditorium: Last names R-Z

1 page of handwritten notes is allowed. Both sides of a 8.5"x11" sheet.

Exam composition: 50% from material after midterm 2, 25% from midterm 1 and 25% from midterm 2.

Please submit a course evaluation **and** an end of term survey

http://algs4.cs.princeton.edu

Course Wrap-Up

- *Intractable problems: P versus NP*
- *Course goals*
- *What you can do next*

Theory of Algorithms: Classifying Problems

Goal. Classify problems according to computational requirements.

complexity	order of growth	examples
linear	N	*min, max, median, Burrows-Wheeler transform, …*
linearithmic	$N \log N$	*sorting, element distinctness, …*
quadratic	N^2	?
⋮	⋮	⋮
exponential	c^N	?

Frustrating news. Huge number of problems have defied classification.

Theory of Algorithms: Classifying Problems

Goal. Classify problems according to computational requirements.

complexity	order of growth	examples
linear	N	*min, max, median, Burrows-Wheeler transform, …*
linearithmic	$N \log N$	*sorting, element distinctness, …*
M(N)	?	*integer multiplication, division, square root, …*
MM(N)	?	*matrix multiplication, $Ax = b$, least square, determinant, …*
⋮	⋮	⋮
NP-complete	*probably not* N^k	3-SAT, IND-SET, ILP, …

Good news. Can put many problems into equivalence classes.

Complexity zoo

Complexity class. Set of problems sharing some computational property.

https://complexityzoo.uwaterloo.ca

Bad news. Lots of complexity classes (496 animals in zoo).

Problem classifications via reductions

Reductions are important in practice to:

- Design algorithms.
- Design reusable software modules.
  - stacks, queues, priority queues, symbol tables, sets, graphs
  - sorting, regular expressions, suffix arrays
  - MST, shortest paths, maxflow, linear programming
- Determine difficulty of your problem and choose the right tool.

Reductions are important in theory to:

- Design algorithms.
- Establish lower bounds.
- Classify problems according to their computational requirements.

We will look at a special classification of algorithms: P versus NP

Intractability: Bird's-eye view

Worse case runtime is $O(N^k)$ or less for some constant k

Def. A problem is intractable if it can't be solved in polynomial time.

Goal. Prove that a problem is intractable.

Worse case runtime is $O(2^N)$ or worse

Two problems that provably require exponential time.

- Given a constant-size program, does it halt in at most K steps?
- Given N-by-N checkers board position, can the first player force a win?

Alan designed the perfect computer

Frustrating news. Few problems have provably exponential time.

A core problem: satisfiability

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

Ex.

$\neg x_1$	*or*	x_2	*or*	x_3			$=$	*true*
x_1	*or*	$\neg x_2$	*or*	x_3			$=$	*true*
$\neg x_1$	*or*	$\neg x_2$	*or*	$\neg x_3$			$=$	*true*
$\neg x_1$	*or*	$\neg x_2$	*or*		*or*	x_4	$=$	*true*
		$\neg x_2$	*or*	x_3	*or*	x_4	$=$	*true*

instance I

x_1	x_2	x_3	x_4
T	T	F	T

solution S

3-SAT. All equations of this form (with three variables per equation).

Key applications.

- Automatic verification systems for software.
- Electronic design automation (EDA) for hardware.
- Many others

Satisfiability is conjectured to be intractable

Q. How to solve an instance of 3-SAT with N variables?

A. Exhaustive search: try all 2^N truth assignments.

intractable, exponential time

Q. Can we do anything substantially more clever?

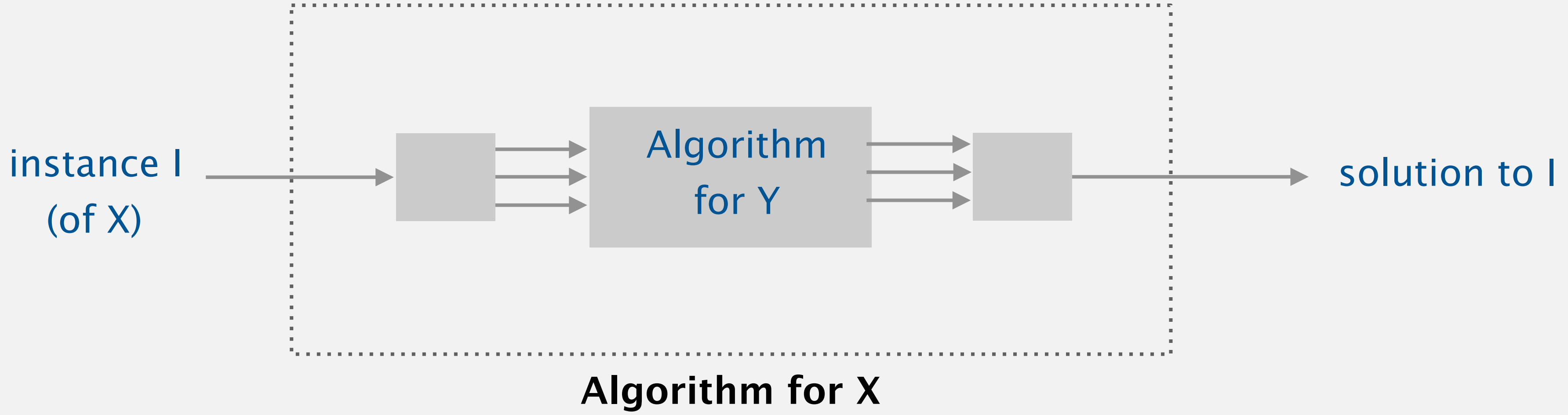
Conjecture (**P** $\neq$ **NP**). 3-SAT is intractable (no poly-time algorithm).

consensus opinion

Polynomial-time reductions

Problem X poly-time reduces to problem Y if X can be solved with:

- Polynomial number of standard computational steps.
- Polynomial number of calls to Y.



Establish intractability. If 3-SAT poly-time reduces to Y, then Y is intractable. (assuming 3-SAT is intractable)

Mentality.

- If I could solve Y in poly-time, then I could also solve 3-SAT in poly-time.
- 3-SAT is believed to be intractable.
- Therefore, so is Y.

Integer linear programming

ILP. Given a system of linear inequalities, find an integral solution.

$$3x_1 + 5x_2 + 2x_3 + x_4 + 4x_5 \geq 10$$

$$5x_1 + 2x_2 + 4x_4 + 1x_5 \leq 7$$

$$x_1 + x_3 + 2x_4 \leq 2$$

$$3x_1 + 4x_3 + 7x_4 \leq 7$$

$$x_1 + x_4 \leq 1$$

$$x_1 + x_3 + x_5 \leq 1$$

$$\text{all } x_i = \{ 0 , 1 \}$$

linear inequalities

integer variables

instance I

x_1	x_2	x_3	x_4	x_5
0	1	0	1	1

solution S

Context. Cornerstone problem in operations research.

Remark. Finding a real-valued solution is tractable (linear programming).

3-SAT poly-time reduces to ILP

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

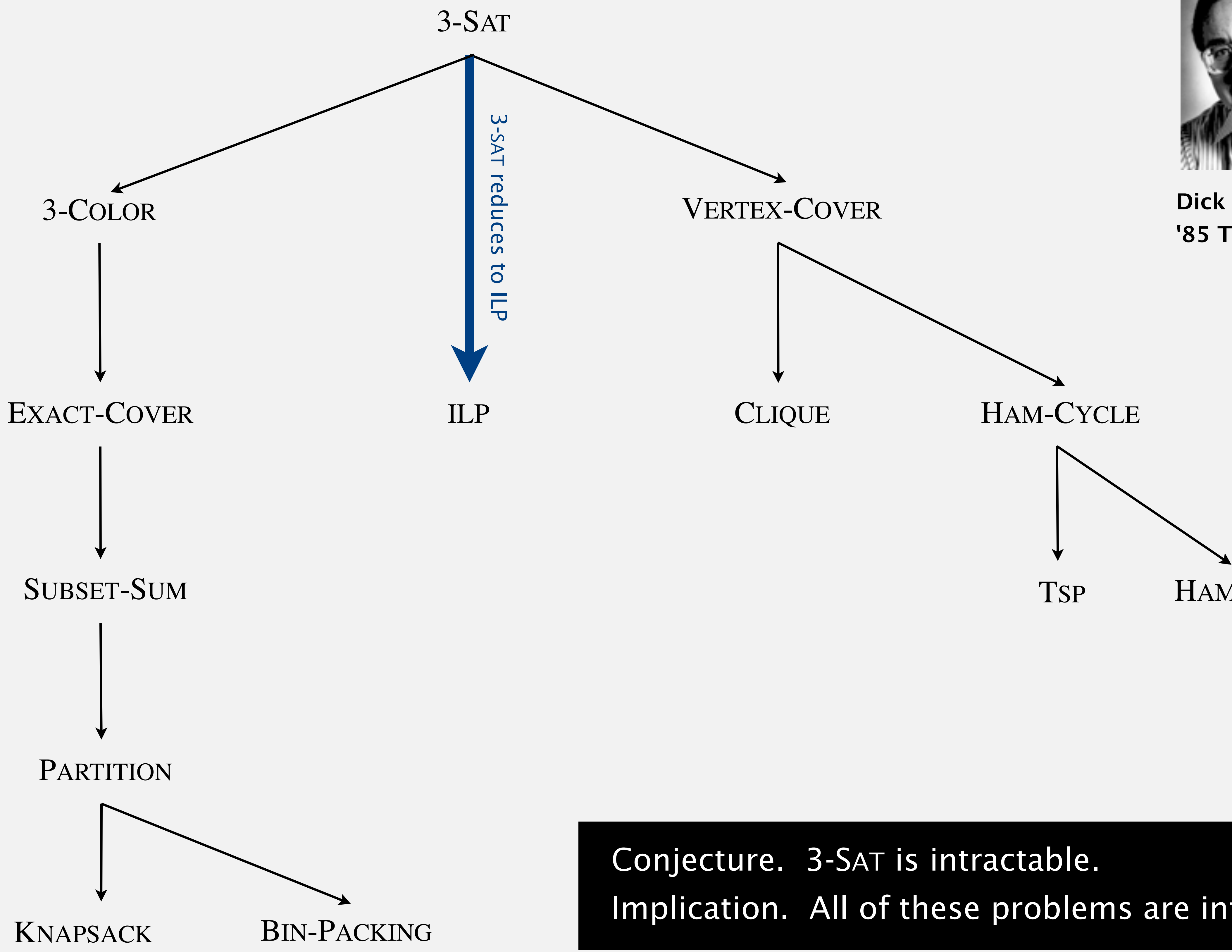
$\neg x_1$	*or*	x_2	*or*	x_3			$=$	*true*
x_1	*or*	$\neg x_2$	*or*	x_3			$=$	*true*
$\neg x_1$	*or*	$\neg x_2$	*or*	$\neg x_3$			$=$	*true*
$\neg x_1$	*or*	$\neg x_2$	*or*		*or*	x_4	$=$	*true*
		$\neg x_2$	*or*	x_3	*or*	x_4	$=$	*true*

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

$(1 - x_1)$	$+$	x_2	$+$	x_3			$\geq$	1
x_1	$+$	$(1 - x_2)$	$+$	x_3			$\geq$	1
$(1 - x_1)$	$+$	$(1 - x_2)$	$+$	$(1 - x_3)$			$\geq$	1
$(1 - x_1)$	$+$	$(1 - x_2)$	$+$		$+$	x_4	$\geq$	1
		$(1 - x_2)$	$+$	x_3	$+$	x_4	$\geq$	1

solution to this ILP instance gives solution to original 3-SAT instance

More poly-time reductions from 3-satisfiability



Dick Karp
'85 Turing award

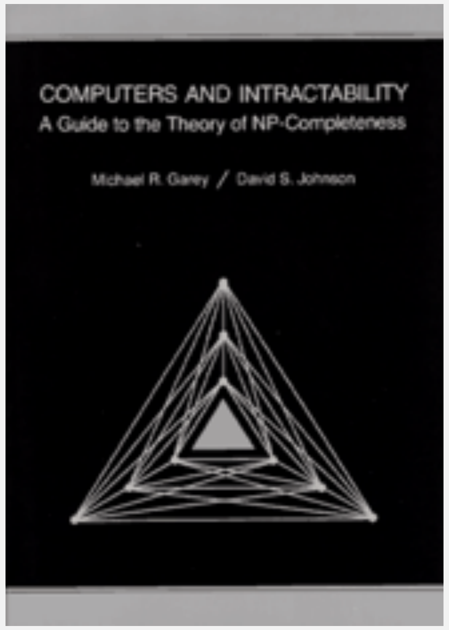
Conjecture. 3-SAT is intractable.
Implication. All of these problems are intractable.

Implications of poly-time reductions from 3-satisfiability

Establishing intractability through poly-time reduction is an important tool in guiding algorithm design efforts.

Q. How to convince yourself that a new problem is (probably) intractable?
A1. [hard way] Long futile search for an efficient algorithm (as for 3-SAT).
A2. [easy way] Reduction from 3-SAT.

Caveat. Intricate reductions are common.



Search problems

Search problem. Problem where you can check a solution in poly-time.

Ex 1. 3-SAT.

$\neg x_1$	*or*	x_2	*or*	x_3			$=$	*true*
x_1	*or*	$\neg x_2$	*or*	x_3			$=$	*true*
$\neg x_1$	*or*	$\neg x_2$	*or*	$\neg x_3$			$=$	*true*
$\neg x_1$	*or*	$\neg x_2$	*or*		*or*	x_4	$=$	*true*
		$\neg x_2$	*or*	x_3	*or*	x_4	$=$	*true*

instance I

x_1	x_2	x_3	x_4
T	T	F	T

solution S

Ex 2. FACTOR. Given an N-bit integer x, find a nontrivial factor.

147573952589676412927

instance I

193707721

solution S

P vs. NP

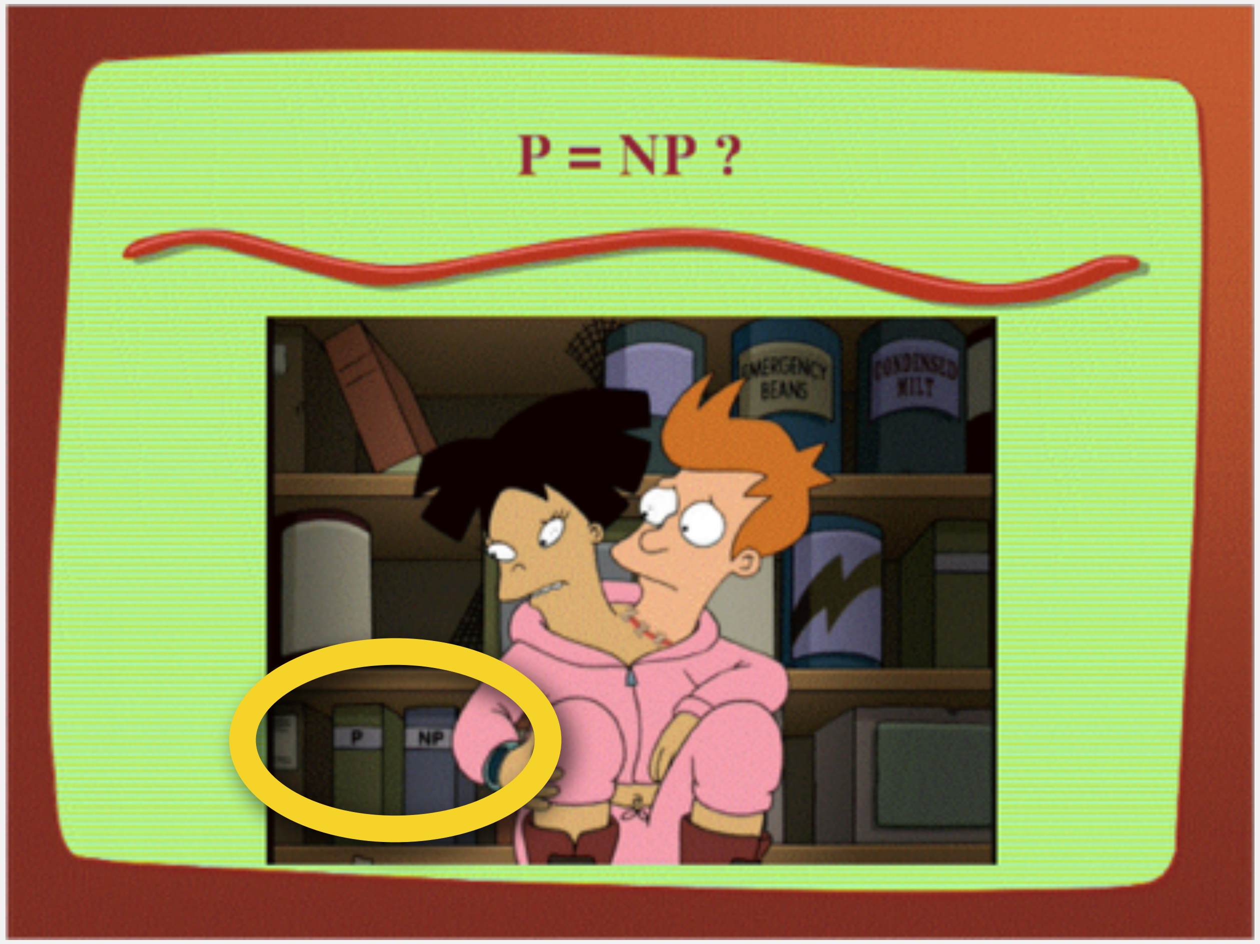
P. Set of search problems solvable in poly-time.

Importance. What scientists and engineers can compute feasibly.

NP. Set of search problems (checkable in poly-time).

Importance. What scientists and engineers aspire to compute feasibly.

Fundamental question.



Consensus opinion. No.

Cook-Levin theorem

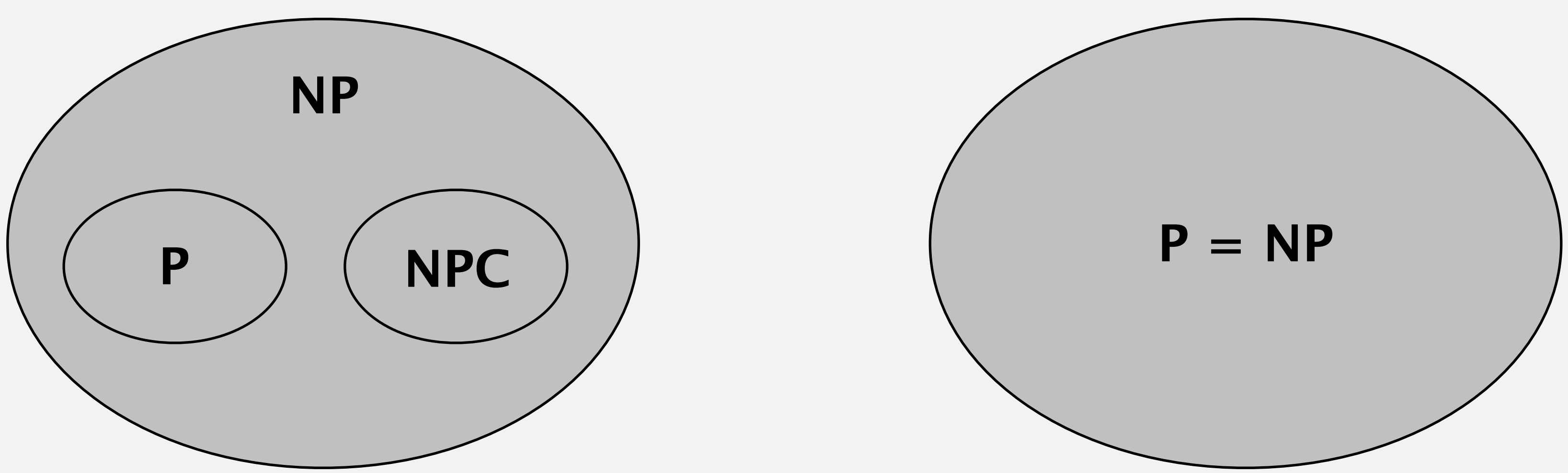
A problem is **NP-COMPLETE** if

- It is in **NP**.
- All problems in **NP** poly-time to reduce to it.

Cook-Levin theorem. 3-SAT is **NP-COMPLETE**.

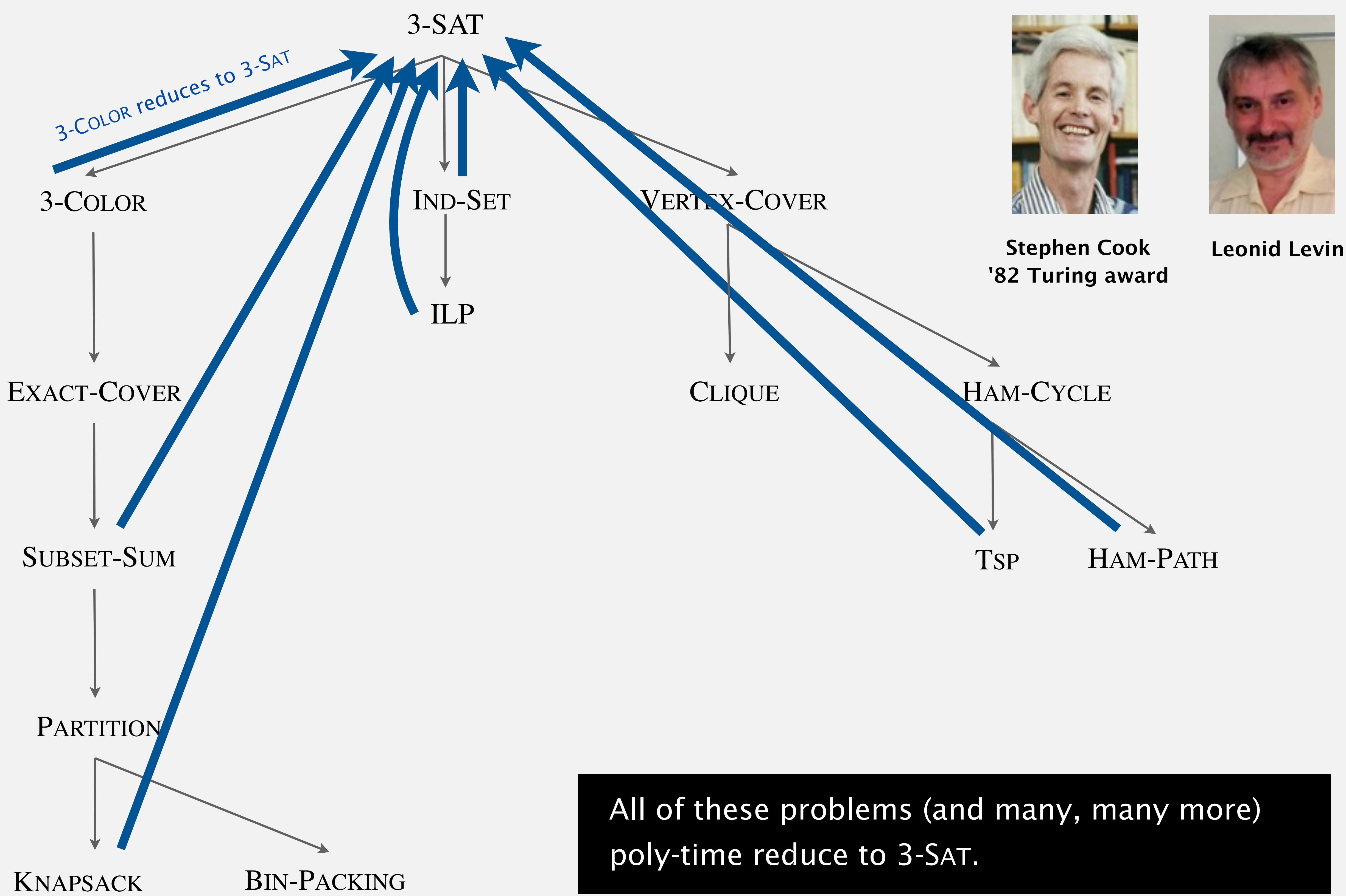
Corollary. 3-SAT is tractable if and only if **P = NP**.

Two worlds.



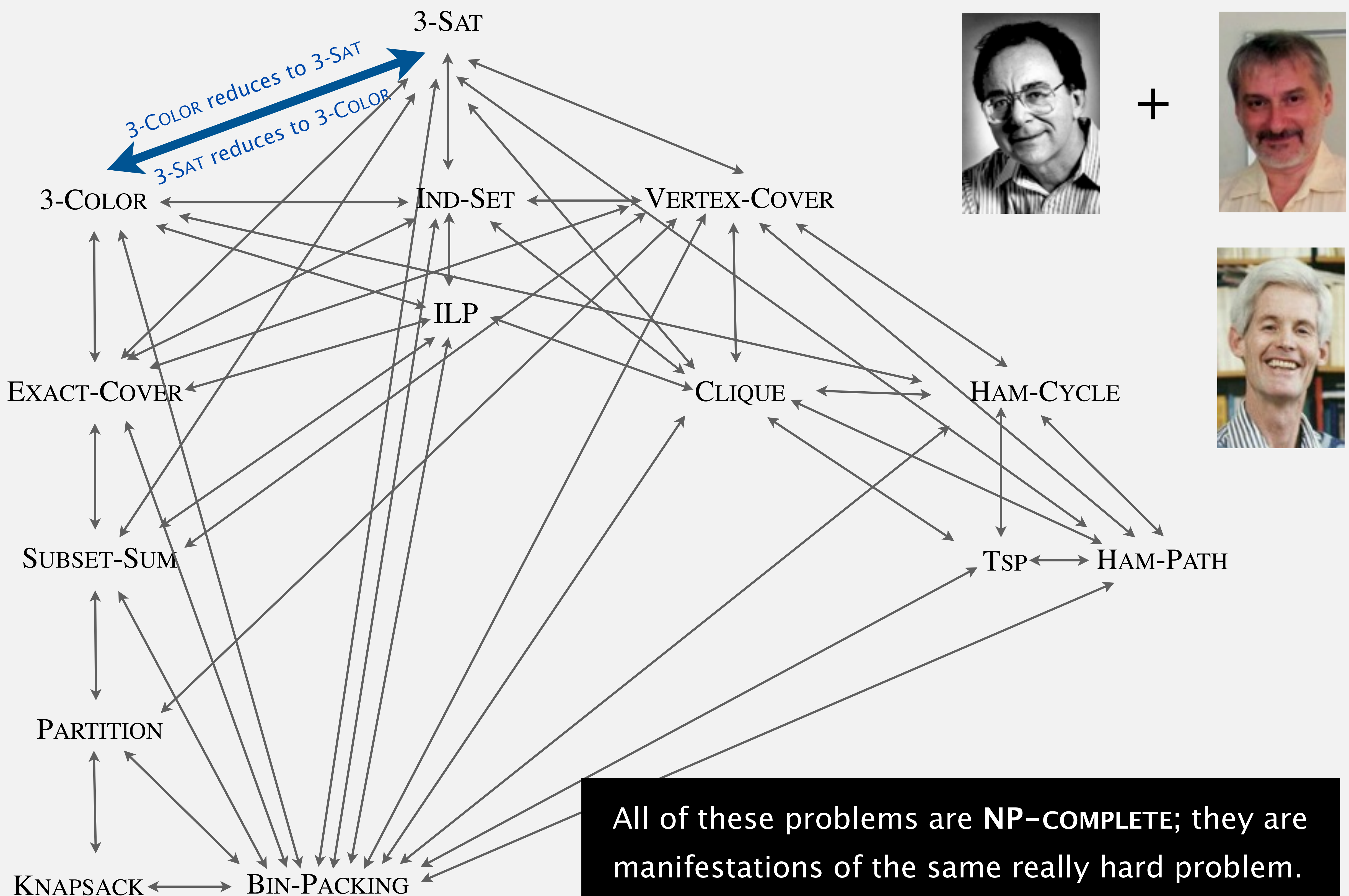
P ≠ NP **P = NP**

Implications of Cook-Levin theorem



All of these problems (and many, many more) poly-time reduce to 3-SAT.

Implications of Karp + Cook-Levin



All of these problems are **NP-COMPLETE**; they are manifestations of the same really hard 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. Problem with no poly-time algorithm.

Many fundamental problems are NP-Complete.

Use theory as 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.

Farewell to Algorithms

What you learned

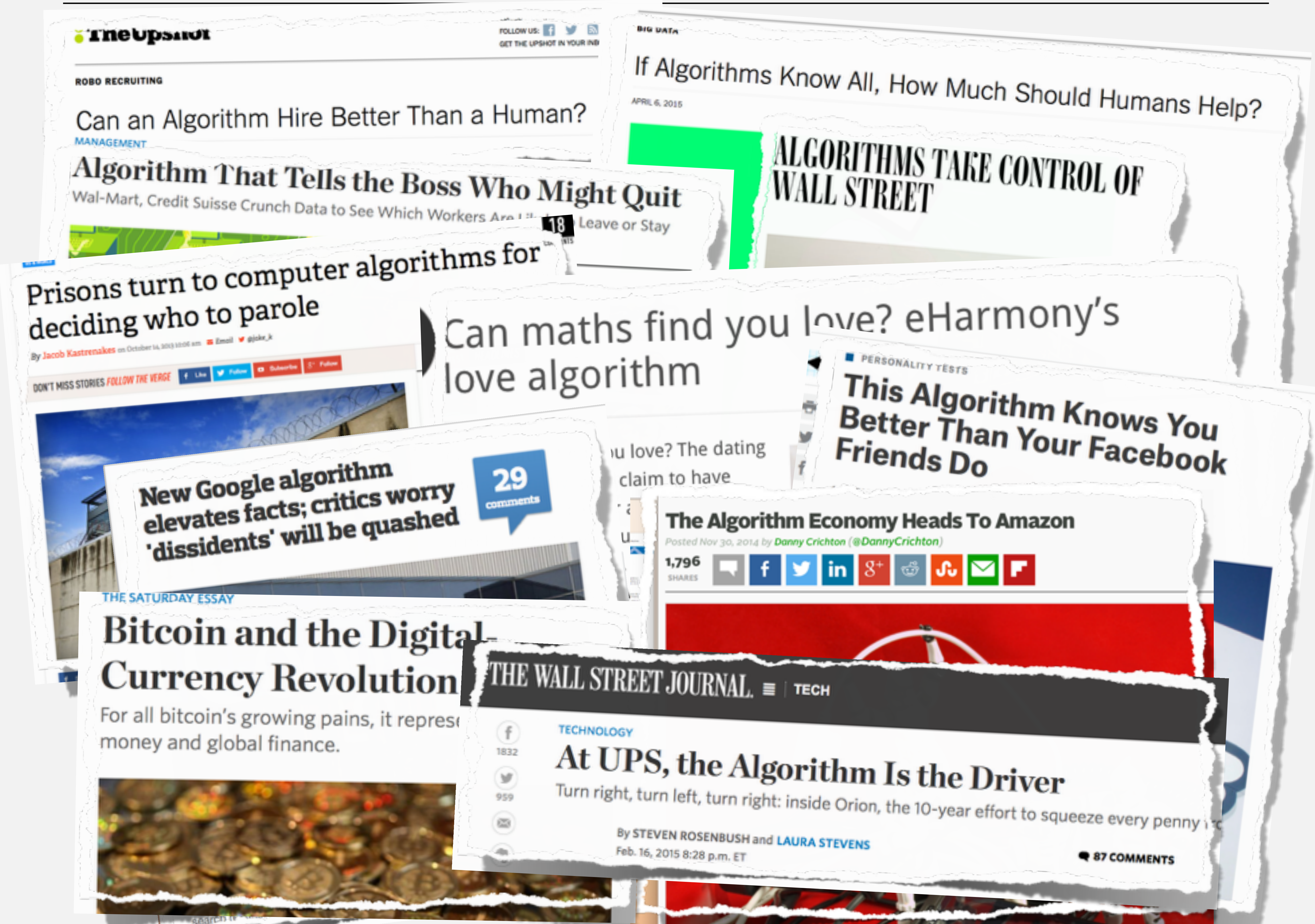
topic	data structures and algorithms
data types	stack, queue, bag, union-find, priority queue
sorting	quicksort, mergesort, heapsort, radix sorts
searching	BST, red-black BST, hash table
graphs	BFS, DFS, Prim, Kruskal, Dijkstra
strings	KMP, regular expressions, tries, data compression

Farewell to Algorithms

What you can do next

CIS 320 - Introduction to Algorithms
CIS 330 - Design Principles of Information Systems
CIS 331 - Intro to Networks and Security
CIS 341 - Compilers and Interpreters
CIS 350 - Software Design/Engineering
CIS 390 - Robotics
CIS 391 - Introduction to Artificial Intelligence
CIS 450 - Database and Information Systems
CIS 460 - Computer Graphics
CIS 500 - Software Foundations
CIS 519 - Introduction to Machine Learning

Algorithms pervade the modern world

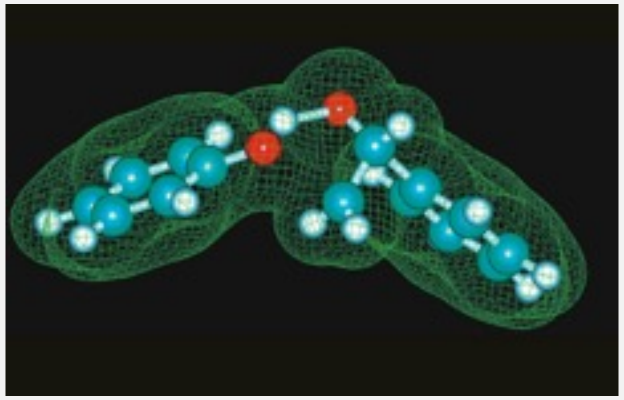


Algorithms are integral to disciplines beyond computer science

You could help to unlock the secrets of life and of the universe.

> *" Computer models mirroring real life have become crucial for most advances made in chemistry today…. Today the computer is just as important a tool for chemists as the test tube. "*
>
> — *Royal Swedish Academy of Sciences*
> *(Nobel Prize in Chemistry 2013)*

Martin Karplus, Michael Levitt, and Arieh Warshel



You are now more proficient programmers

" I will, in fact, claim that the difference between a bad programmer and a good one is whether the programer considers code or data structures more important. Bad programmers worry about the code. Good programmers worry about data structures and their relationships. "

— Linus Torvalds (creator of Linux)

You are now better prepared for your job interviews

Farewell to Algorithms

My hope for what you got out of 121:

- You are now a more proficient programmer
- You have a toolkit of algorithms that you can use in your programs
- You have an understanding of why selecting one data structure or algorithm over another is advantageous for large data sets
- You have confidence going into computer science job interviews

Thank you!

I really enjoyed teaching you!

Please stay in touch!

- Email me if you use 121 knowledge to get a job. ccb@upenn.edu
- Take my classes. NETS 213 "Crowdsourcing and Human Computation" (this Spring), CIS 525 "Machine Translation" (next Spring)
- Learn about my research. Stop by my office (Levine 506). I'm always eager to talk about my research and explore potential undergraduate research opportunities.

Thank you for making this such a good experience for me.