



<http://algs4.cs.princeton.edu>

## 6.5 REDUCTIONS

---

- ▶ *introduction*
- ▶ *designing algorithms*
- ▶ *establishing lower bounds*
- ▶ *classifying problems*
- ▶ *intractability*



<http://algs4.cs.princeton.edu>

## 6.5 REDUCTIONS

---

- ▶ *introduction*
- ▶ *designing algorithms*
- ▶ *establishing lower bounds*
- ▶ *classifying problems*
- ▶ *intractability*

# Bird's-eye view

---

**Goal.** Classify **problems** according to computational requirements.

complexity	order of growth	examples
<b>linear</b>	$N$	<i>min, max, median, Burrows-Wheeler transform, ...</i>
<b>linearithmic</b>	$N \log N$	<i>sorting, element distinctness, closest pair, Euclidean MST, ...</i>
<b>quadratic</b>	$N^2$	?
$\vdots$	$\vdots$	$\vdots$
<b>exponential</b>	$c^N$	?

**Frustrating news.** Huge number of problems have defied classification.

# Bird's-eye view

---

**Goal.** Classify **problems** according to computational requirements.

**Goal 2.** Suppose we could (could not) solve problem  $X$  efficiently.  
What else could (could not) we solve efficiently?

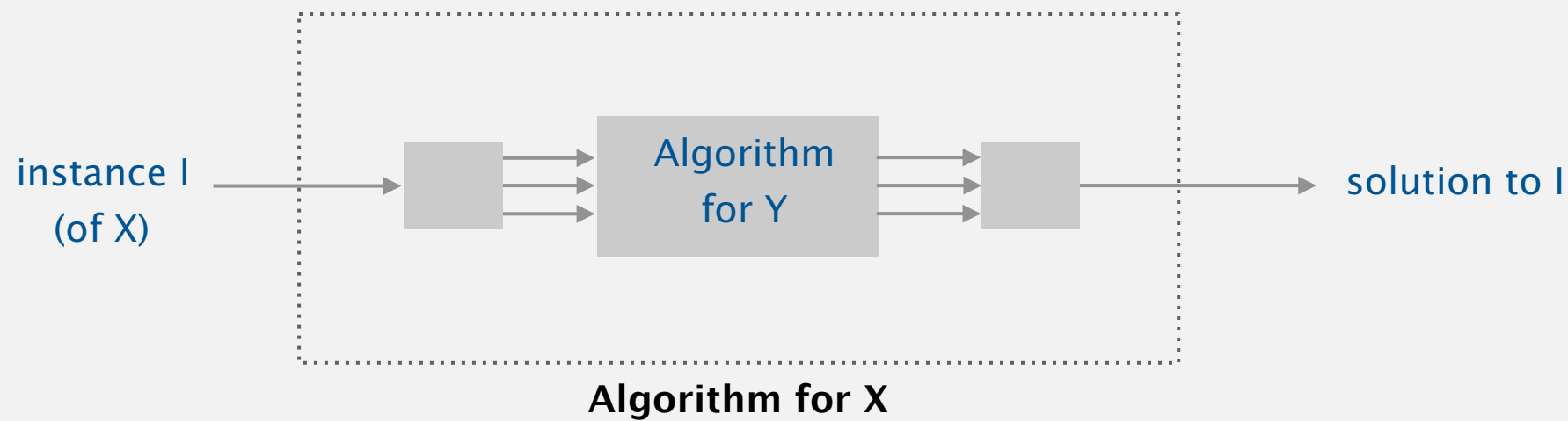


*“ Give me a lever long enough and a fulcrum on which to place it, and I shall move the world. ” — Archimedes*

# Reduction

---

**Def.** Problem  $X$  **reduces to** problem  $Y$  if you can use an algorithm that solves  $Y$  to help solve  $X$ .



Cost of solving  $X$  = total cost of solving  $Y$  + cost of reduction.

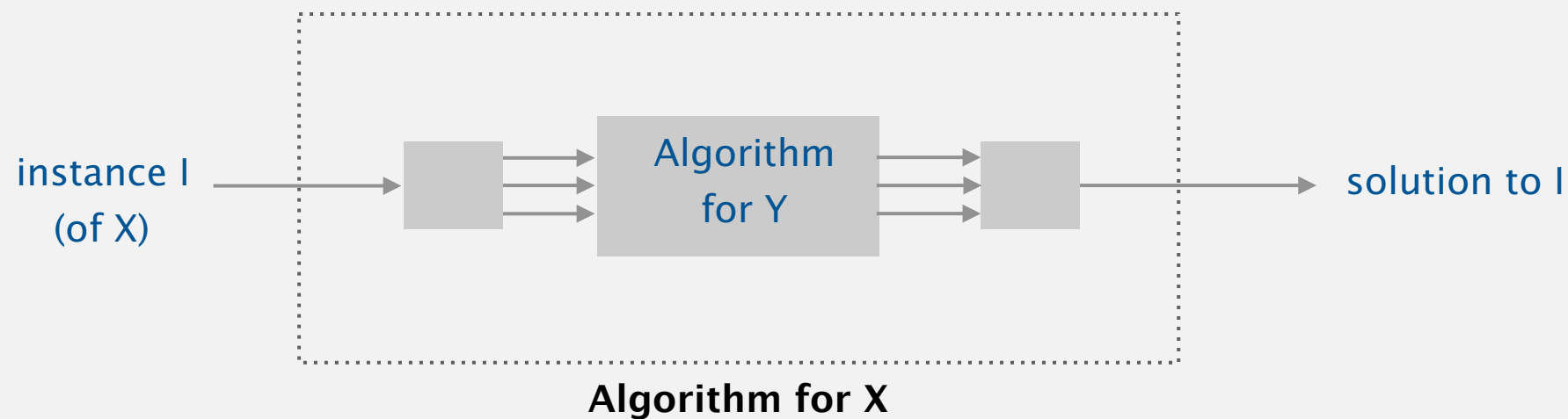
↑  
perhaps many calls to  $Y$   
on problems of different sizes  
(typically only 1 call)

↑  
preprocessing and postprocessing  
(typically less than cost of solving  $Y$ )

# Reduction

---

**Def.** Problem  $X$  **reduces to** problem  $Y$  if you can use an algorithm that solves  $Y$  to help solve  $X$ .



**Ex 1.** [finding the median reduces to sorting]

To find the median of  $N$  items:

- Sort  $N$  items.
- Return item in the middle.

**Cost of finding the median.**  $N \log N + 1$ .

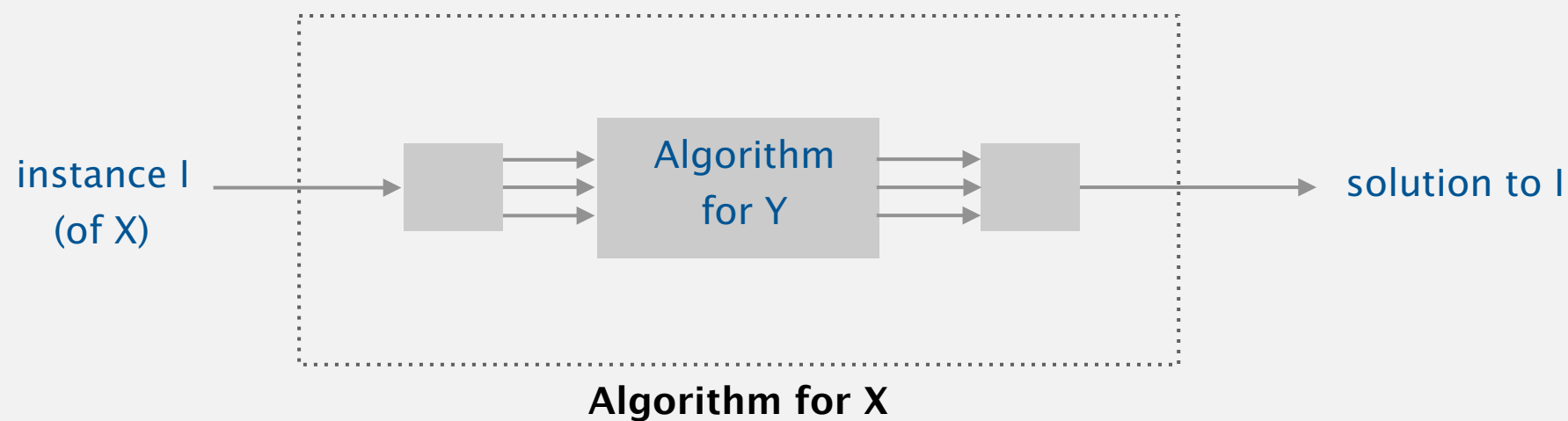
cost of sorting

cost of reduction

# Reduction

---

**Def.** Problem  $X$  **reduces to** problem  $Y$  if you can use an algorithm that solves  $Y$  to help solve  $X$ .



**Ex 2.** [element distinctness reduces to sorting]

To solve element distinctness on  $N$  items:

- Sort  $N$  items.
- Check adjacent pairs for equality.

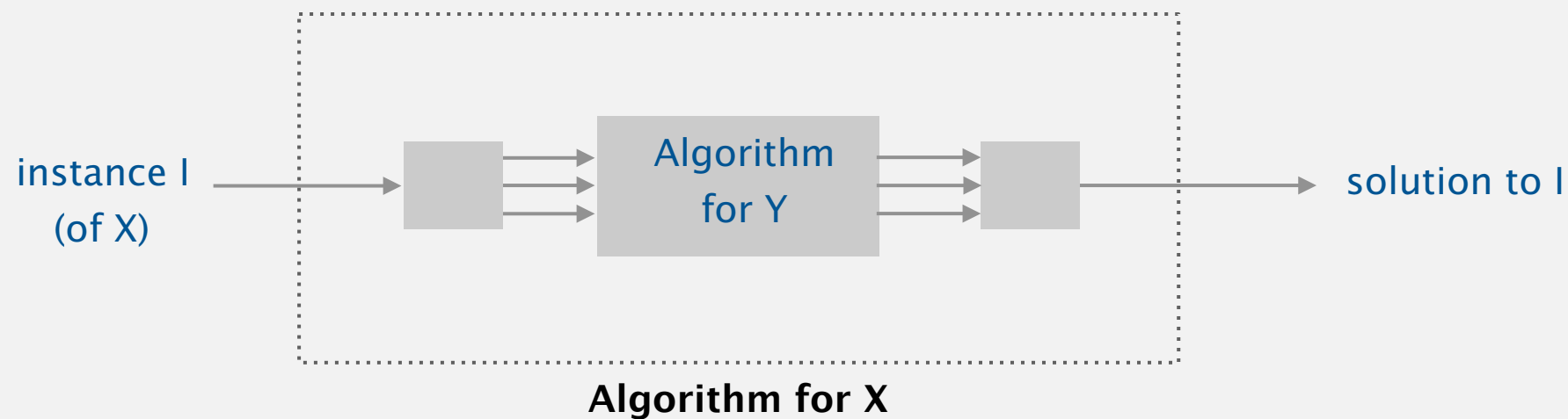
**Cost of element distinctness.**  $N \log N + N$ .

Two red arrows point to the terms in the equation: one from "cost of sorting" to  $N \log N$ , and another from "cost of reduction" to  $N$ .

# Reduction

---

**Def.** Problem  $X$  **reduces to** problem  $Y$  if you can use an algorithm that solves  $Y$  to help solve  $X$ .



**Novice error.** Confusing  $X$  reduces to  $Y$  with  $Y$  reduces to  $X$ .

ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO  
THE ENTSCHEIDUNGSPROBLEM

*By* A. M. TURING.

[Received 28 May, 1936.—Read 12 November, 1936.]







<http://algs4.cs.princeton.edu>

## 6.5 REDUCTIONS

---

- ▶ *introduction*
- ▶ *designing algorithms*
- ▶ *establishing lower bounds*
- ▶ *classifying problems*
- ▶ *intractability*

# Reduction: design algorithms

---

**Def.** Problem  $X$  **reduces to** problem  $Y$  if you can use an algorithm that solves  $Y$  to help solve  $X$ .

**Design algorithm.** Given an algorithm for  $Y$ , can also solve  $X$ .

**More familiar reductions.**

- Mincut reduces to maxflow.
- Arbitrage reduces to negative cycles.
- Bipartite matching reduces to maxflow.
- Seam carving reduces to shortest paths in a DAG.
- Burrows-Wheeler transform reduces to suffix sort.
- ...

**Mentality.** Since I know how to solve  $Y$ , can I use that algorithm to solve  $X$ ?

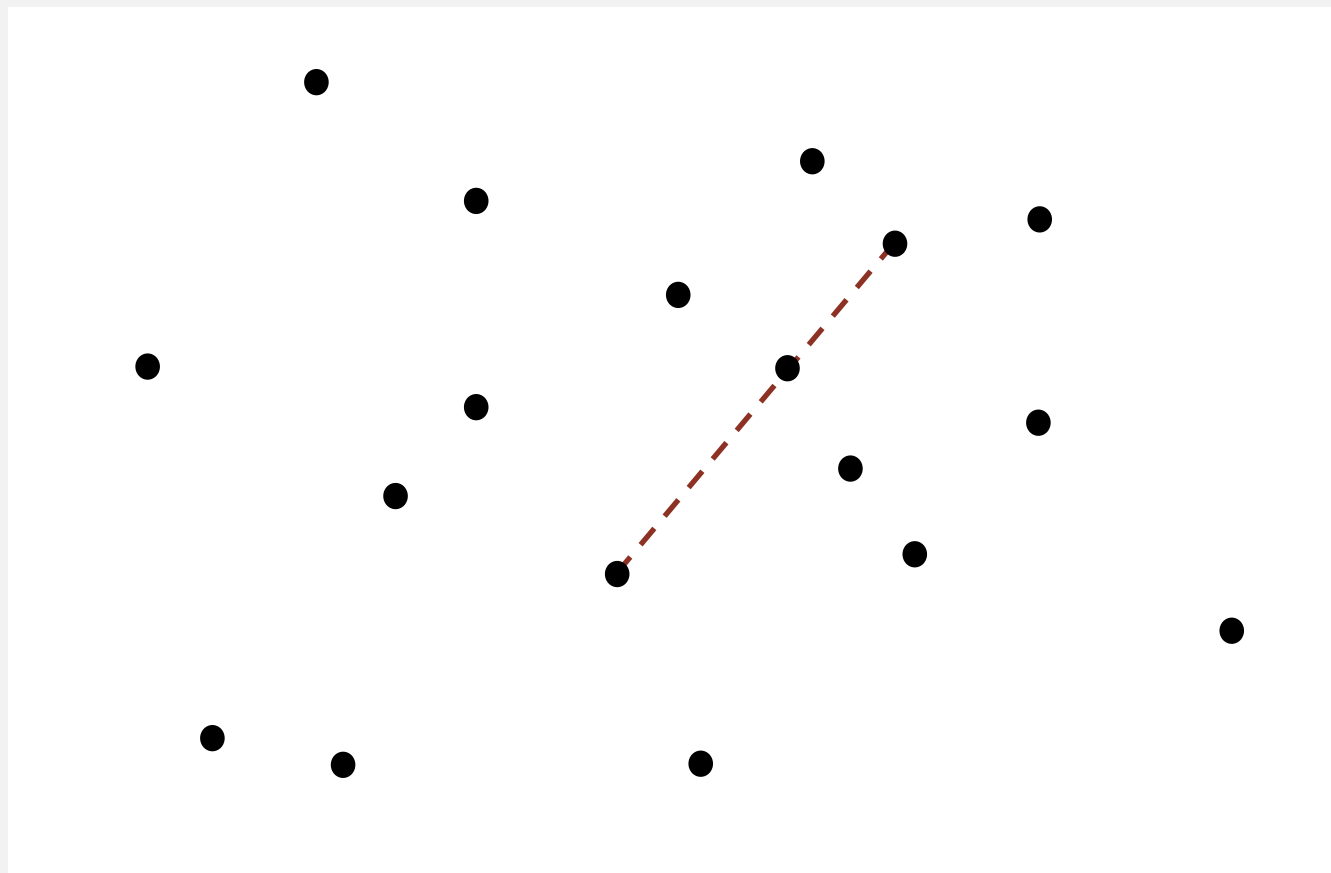


programmer's version: I have code for  $Y$ . Can I use it for  $X$ ?

## 3-collinear

---

**3-COLLINEAR.** Given  $N$  distinct points in the plane, are there 3 (or more) that all lie on the same line?



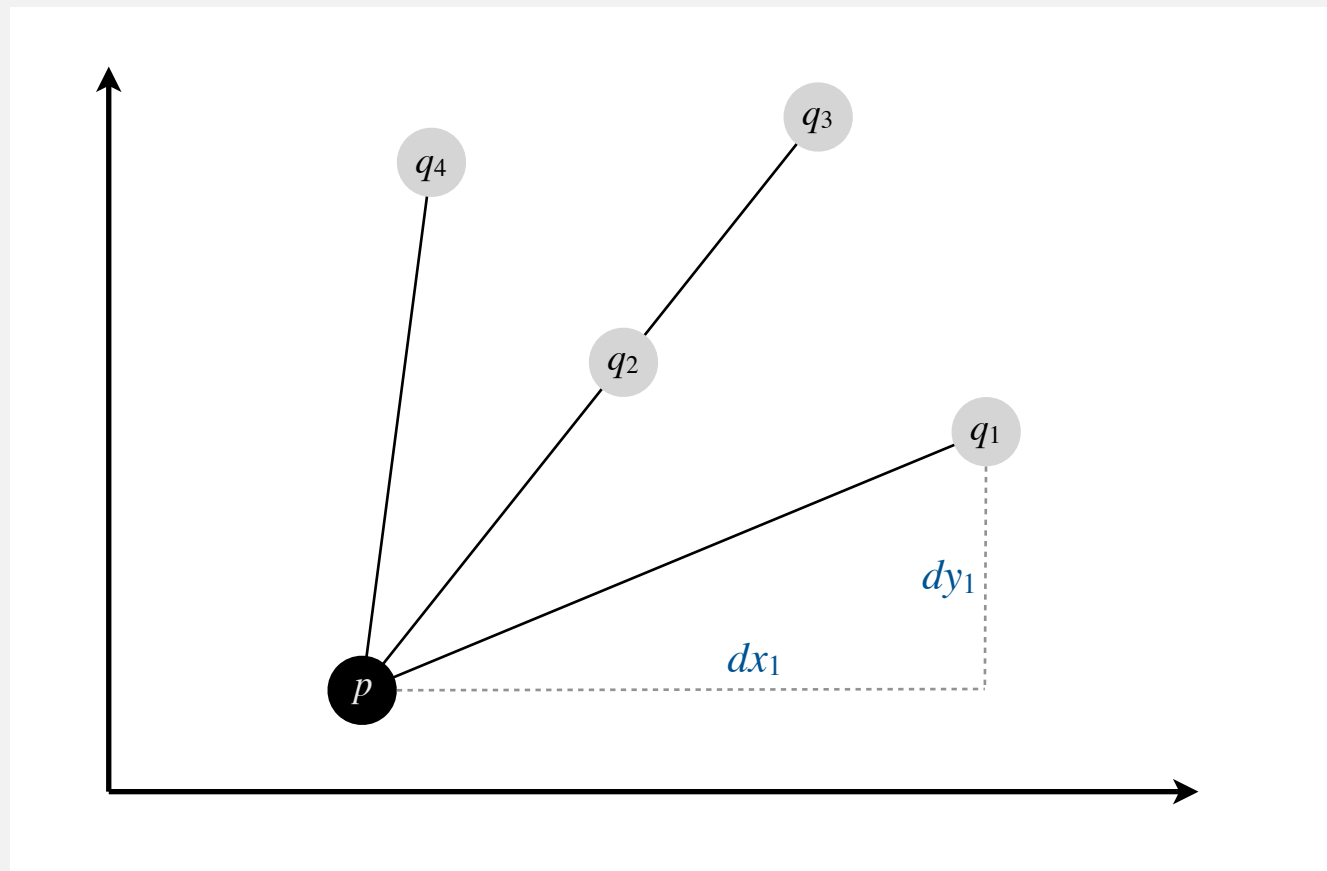
3-collinear

**Brute force  $N^3$ .** For all triples of points  $(p, q, r)$ , check if they are collinear.

# 3-collinear reduces to sorting

**Sorting-based algorithm.** For each point  $p$ ,

- Compute the slope that each other point  $q$  makes with  $p$ .
- Sort the  $N - 1$  points by slope.
- Collinear points are adjacent.

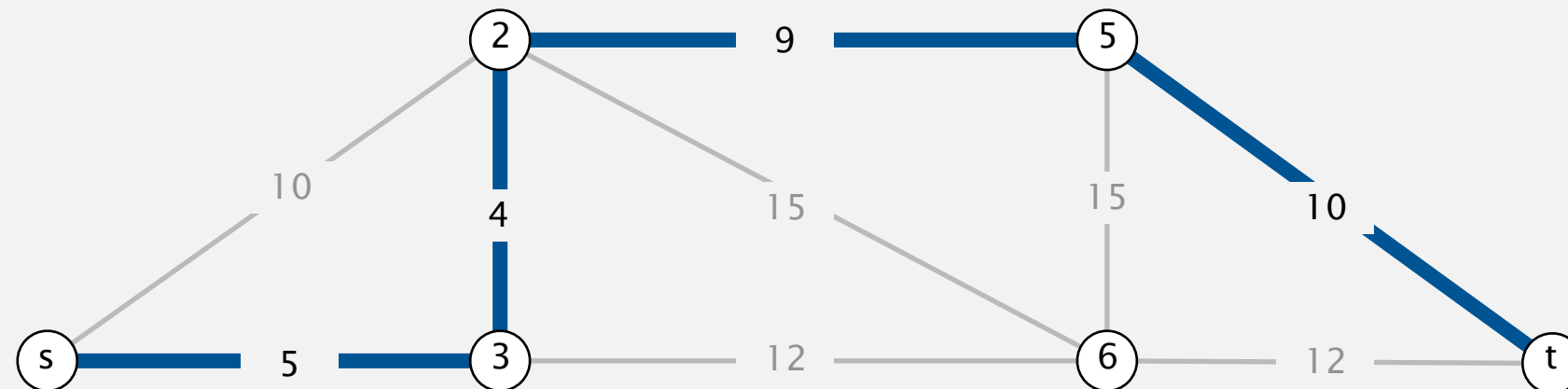


**Cost of solving 3-COLLINEAR.**  $N^2 \log N + N^2$ .

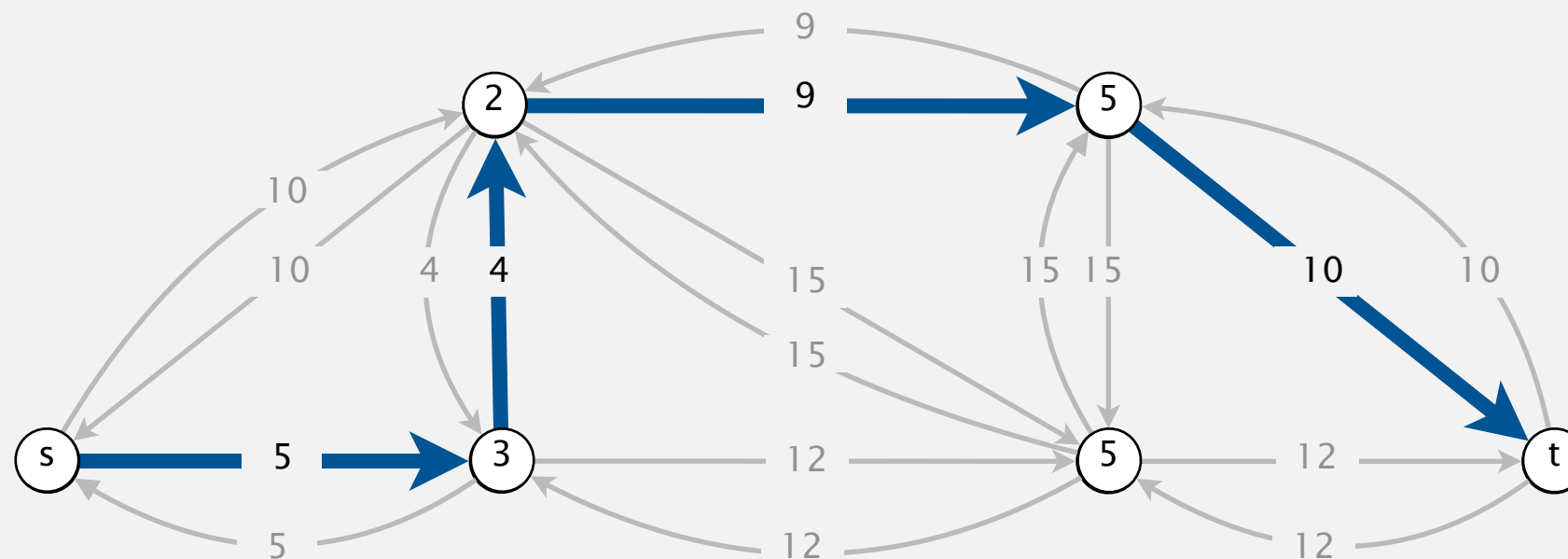
cost of sorting (N times)  $\swarrow$   $N^2 \log N$   $\nwarrow$  cost of reduction  $N^2$

# Shortest paths on edge-weighted graphs and digraphs

**Proposition.** Undirected shortest paths (with nonnegative weights) reduces to directed shortest path.



**Pf.** Replace each undirected edge by two directed edges.



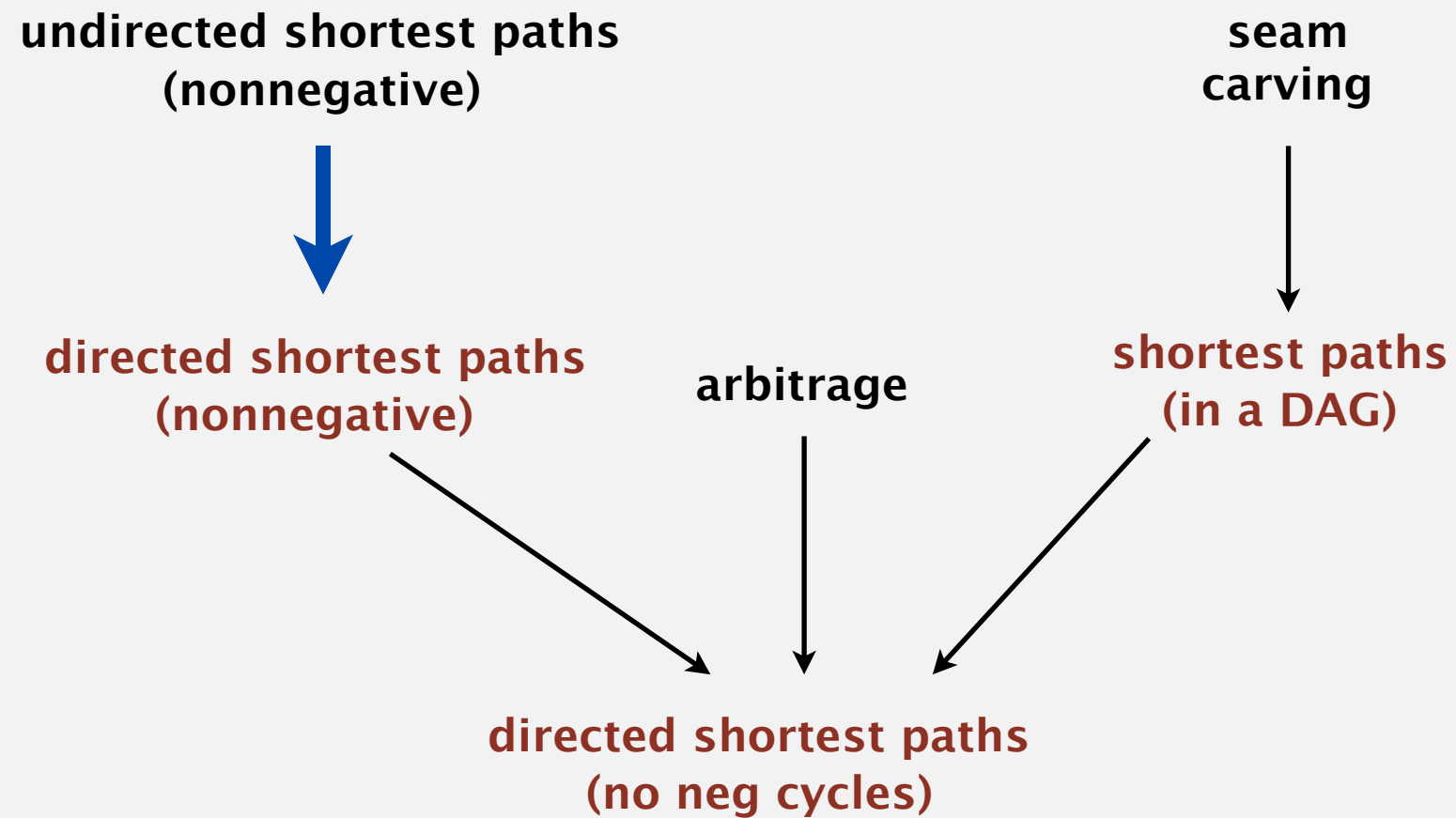
**Cost of solving undirected shortest paths.**  $E \log V + (E + V)$ .

cost of Dijkstra

cost of reduction

# Some reductions in combinatorial optimization

---





<http://algs4.cs.princeton.edu>

## 6.5 REDUCTIONS

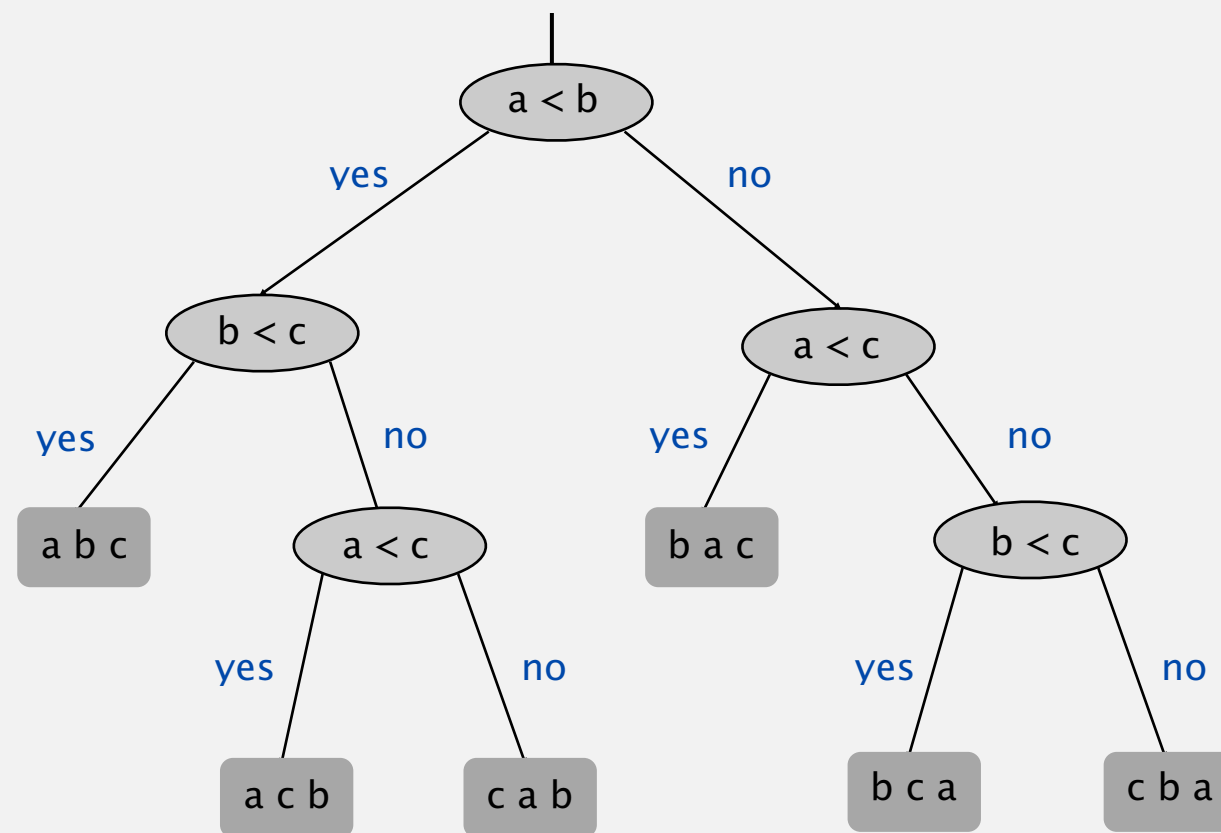
---

- ▶ *introduction*
- ▶ *designing algorithms*
- ▶ *establishing lower bounds*
- ▶ *classifying problems*
- ▶ *intractability*

# Bird's-eye view

**Goal.** Prove that a problem requires a certain number of steps.

**Ex.** In decision tree model, any compare-based sorting algorithm requires  $\Omega(N \log N)$  compares in the worst case.



argument must apply to all conceivable algorithms

**Bad news.** Very difficult to establish lower bounds from scratch.

**Good news.** Spread  $\Omega(N \log N)$  lower bound to  $Y$  by reducing sorting to  $Y$ .

assuming cost of reduction is not too high



# Linear-time reductions

---

**Def.** Problem  $X$  **linear-time reduces** to problem  $Y$  if  $X$  can be solved with:

- Linear number of standard computational steps.
- Constant number of calls to  $Y$ .

**Establish lower bound:**

- If  $X$  takes  $\Omega(N \log N)$  steps, then so does  $Y$ .
- If  $X$  takes  $\Omega(N^2)$  steps, then so does  $Y$ .

**Mentality.**

- If I could easily solve  $Y$ , then I could easily solve  $X$ .
- I can't easily solve  $X$ .
- Therefore, I can't easily solve  $Y$ .

# ELEMENT-DISTINCTNESS linear-time reduces to 2D-CLOSEST-PAIR

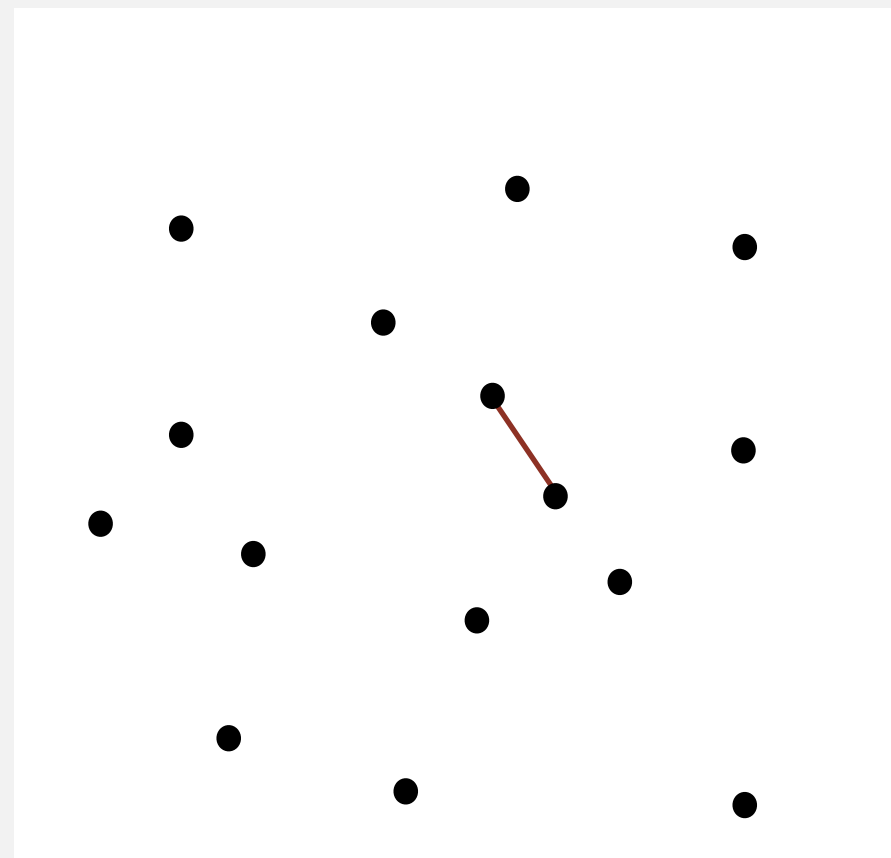
---

**ELEMENT-DISTINCTNESS.** Given  $N$  elements, are any two equal?

**2D-CLOSEST-PAIR.** Given  $N$  points in the plane, find the closest pair.

```
590584
-23439854
1251432
-2861534
3988818
-43434213
333255
13546464
89885444
-43434213
11998833
```

element distinctness



2d closest pair

# ELEMENT-DISTINCTNESS linear-time reduces to 2D-CLOSEST-PAIR

---

**ELEMENT-DISTINCTNESS.** Given  $N$  elements, are any two equal?

**2D-CLOSEST-PAIR.** Given  $N$  points in the plane, find the closest pair.

**Proposition.** ELEMENT-DISTINCTNESS linear-time reduces to 2D-CLOSEST-PAIR.

**Pf.**

- ELEMENT-DISTINCTNESS instance:  $x_1, x_2, \dots, x_N$ .
- 2D-CLOSEST-PAIR instance:  $(x_1, x_1), (x_2, x_2), \dots, (x_N, x_N)$ .
- The  $N$  elements are distinct iff distance of closest pair  $> 0$ .

allows linear tests like  $x_i < x_j$

and quadratic tests like  $(x_i - x_k)^2 + (x_j - x_k)^2 > 4$

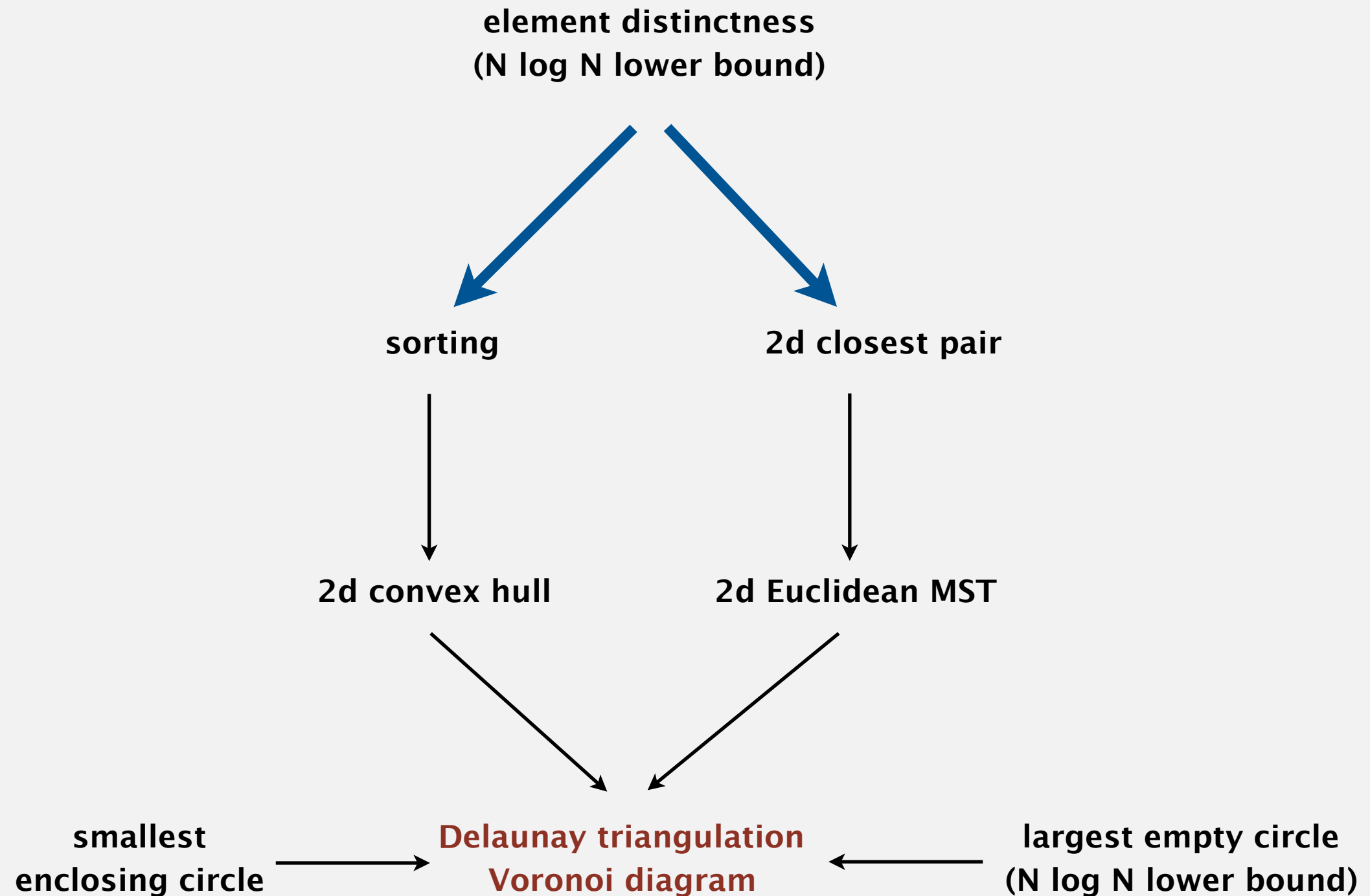


**ELEMENT-DISTINCTNESS lower bound.** In quadratic decision tree model, any algorithm that solves ELEMENT-DISTINCTNESS takes  $\Omega(N \log N)$  steps.

**Implication.** In quadratic decision tree model, any algorithm for 2D-CLOSEST-PAIR takes  $\Omega(N \log N)$  steps.

# Some linear-time reductions in computational geometry

---



# Lower bound for 3-COLLINEAR

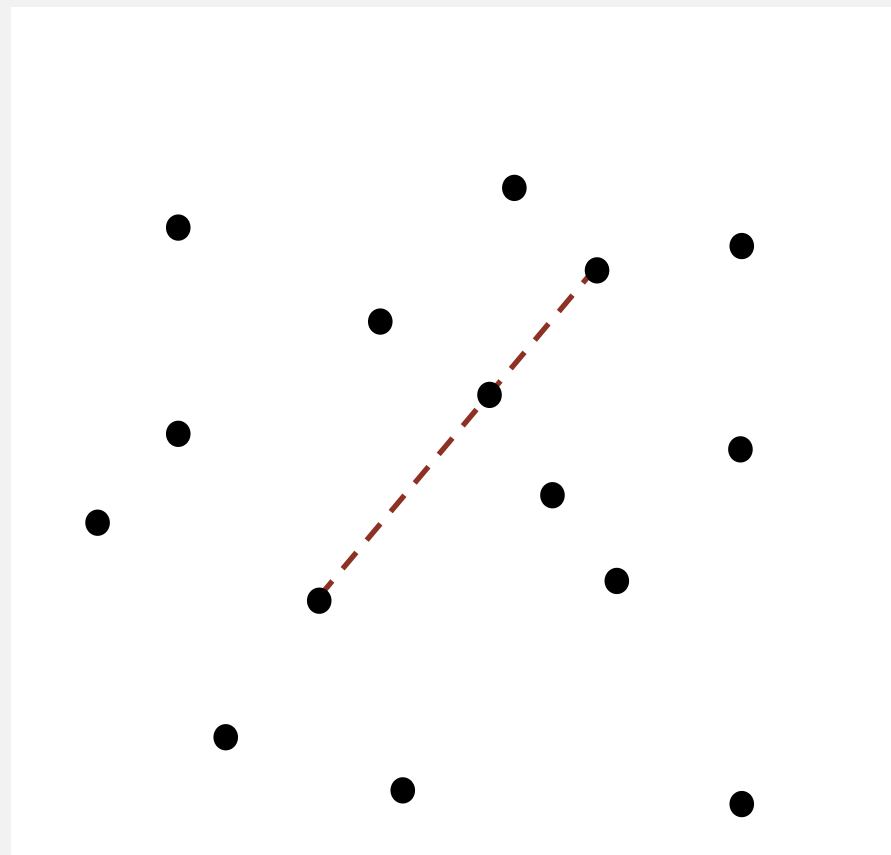
---

**3-SUM.** Given  $N$  distinct integers, are there three that sum to 0?

**3-COLLINEAR.** Given  $N$  distinct points in the plane, are there 3 (or more) that lie on the same line?

590584  
-23439854  
1251432  
-2861534  
3988818  
-4190745  
333255  
13546464  
89885444  
-43434213  
11998833

3-sum



3-collinear

# Lower bound for 3-COLLINEAR

---


**3-SUM.** Given  $N$  distinct integers, are there three that sum to 0?

**3-COLLINEAR.** Given  $N$  distinct points in the plane, are there 3 (or more) that lie on the same line?

**Proposition.** 3-SUM linear-time reduces to 3-COLLINEAR.

**Pf.** [next two slides]

lower-bound mentality:  
if I can't solve 3-SUM in  $N^{1.99}$  time,  
I can't solve 3-COLLINEAR  
in  $N^{1.99}$  time either



**Conjecture.** Any algorithm for 3-SUM requires  $\Omega(N^{2-\epsilon})$  steps.

**Implication.** No sub-quadratic algorithm for 3-COLLINEAR likely.

our  $N^2 \log N$  algorithm was pretty good



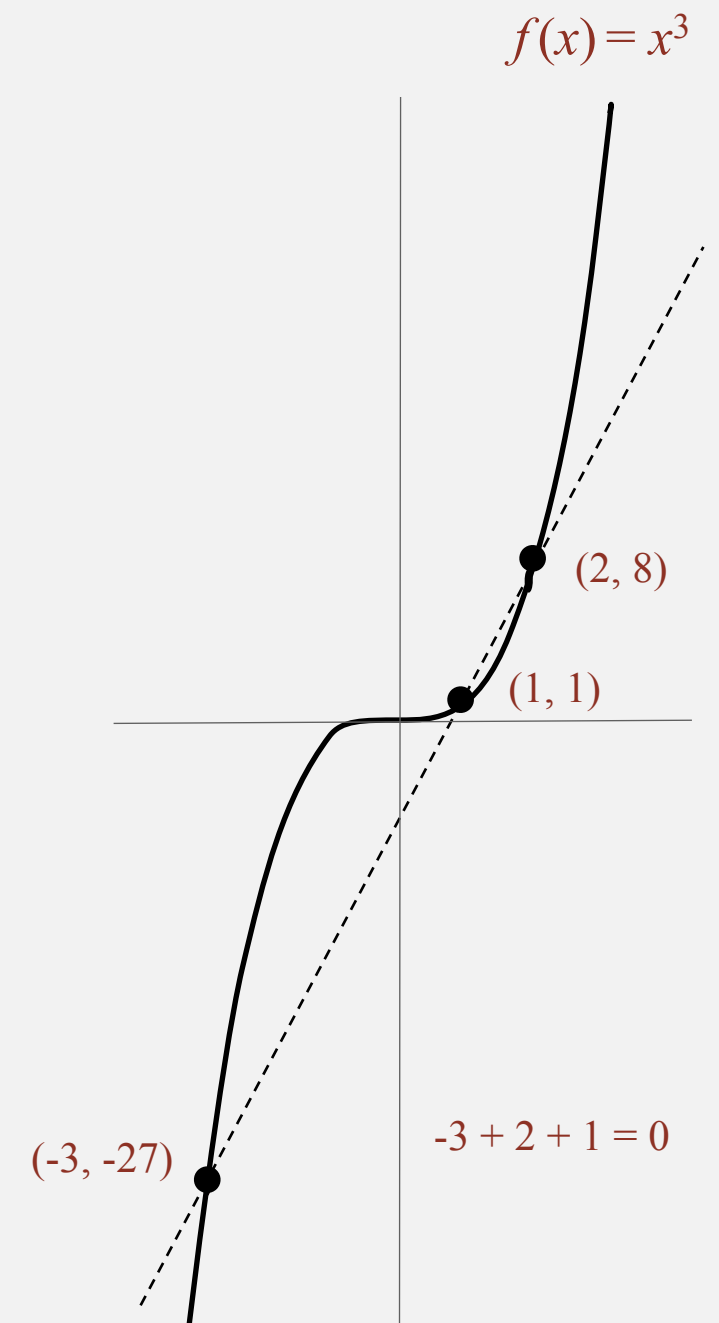
# 3-SUM linear-time reduces to 3-COLLINEAR

---

**Proposition.** 3-SUM linear-time reduces to 3-COLLINEAR.

- 3-SUM instance:  $x_1, x_2, \dots, x_N$ .
- 3-COLLINEAR instance:  $(x_1, x_1^3), (x_2, x_2^3), \dots, (x_N, x_N^3)$ .

**Lemma.** If  $a, b$ , and  $c$  are distinct, then  $a + b + c = 0$  if and only if  $(a, a^3), (b, b^3)$ , and  $(c, c^3)$  are collinear.



## 3-SUM linear-time reduces to 3-COLLINEAR

---

**Proposition.** 3-SUM linear-time reduces to 3-COLLINEAR.

- 3-SUM instance:  $x_1, x_2, \dots, x_N$ .
- 3-COLLINEAR instance:  $(x_1, x_1^3), (x_2, x_2^3), \dots, (x_N, x_N^3)$ .

**Lemma.** If  $a, b$ , and  $c$  are distinct, then  $a + b + c = 0$  if and only if  $(a, a^3), (b, b^3)$ , and  $(c, c^3)$  are collinear.

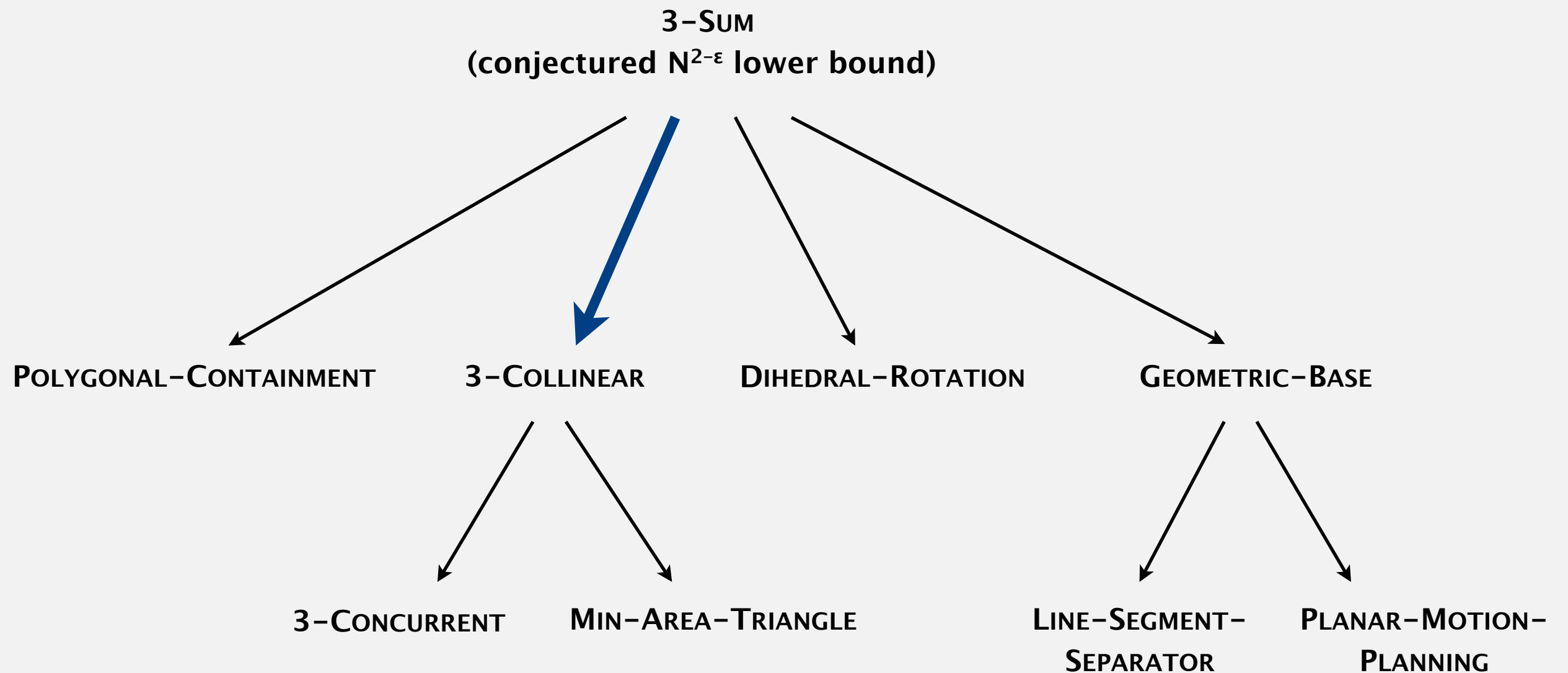
**Pf.** Three distinct points  $(a, a^3), (b, b^3)$ , and  $(c, c^3)$  are collinear iff:

$$\begin{aligned} 0 &= \begin{vmatrix} a & a^3 & 1 \\ b & b^3 & 1 \\ c & c^3 & 1 \end{vmatrix} \\ &= a(b^3 - c^3) - b(a^3 - c^3) + c(a^3 - b^3) \\ &= (a - b)(b - c)(c - a)(a + b + c) \end{aligned}$$



# More geometric reductions and lower bounds

---



# Complexity of 3-SUM

---

April 2014. Some recent evidence that the complexity might be  $N^{3/2}$ .

## Threesomes, Degenerates, and Love Triangles\*

Allan Grønlund  
MADALGO, Aarhus University

Seth Pettie  
University of Michigan

April 4, 2014

### Abstract

The 3SUM problem is to decide, given a set of  $n$  real numbers, whether any three sum to zero. We prove that the decision tree complexity of 3SUM is  $O(n^{3/2}\sqrt{\log n})$ , that there is a randomized 3SUM algorithm running in  $O(n^2(\log \log n)^2/\log n)$  time, and a deterministic algorithm running in  $O(n^2(\log \log n)^{5/3}/(\log n)^{2/3})$  time. These results refute the strongest version of the 3SUM conjecture, namely that its decision tree (and algorithmic) complexity is  $\Omega(n^2)$ .

# Establishing lower bounds: summary

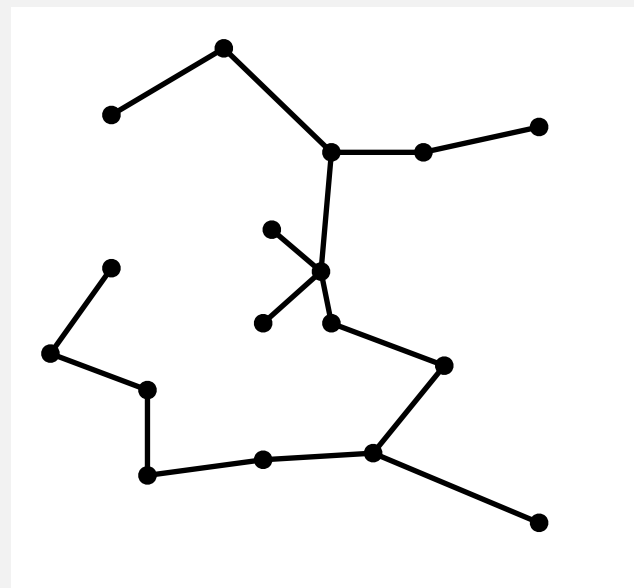
---

Establishing lower bounds through reduction is an important tool in guiding algorithm design efforts.

**Q.** How to convince yourself no linear-time EUCLIDEAN-MST algorithm exists?

**A1.** [hard way] Long futile search for a linear-time algorithm.

**A2.** [easy way] Linear-time reduction from element distinctness.



2d Euclidean MST





<http://algs4.cs.princeton.edu>

## 6.5 REDUCTIONS

---

- ▶ *introduction*
- ▶ *designing algorithms*
- ▶ *establishing lower bounds*
- ▶ *classifying problems*
- ▶ *intractability*

# Classifying problems: summary

---

**Goal.** Problem with algorithm that matches lower bound.

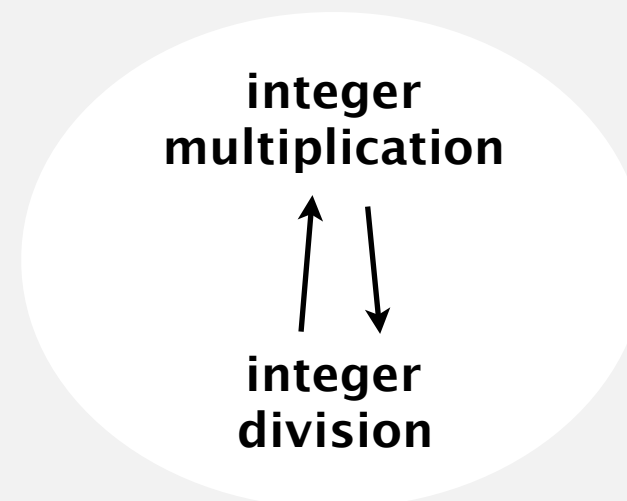
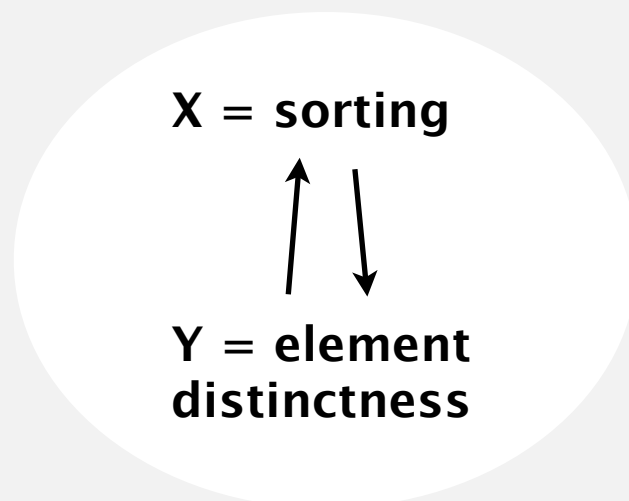
**Ex.** Sorting and element distinctness have complexity  $N \log N$ .

**Goal'.** Prove that two problems  $X$  and  $Y$  have the same complexity.

First, show that problem  $X$  linear-time reduces to  $Y$ .

- Second, show that  $Y$  linear-time reduces to  $X$ .
- Conclude that  $X$  has complexity  $N^b$  iff  $Y$  has complexity  $N^b$  for  $b \geq 1$ .

even if we don't know what it is





# Integer arithmetic reductions

---

**Integer multiplication.** Given two  $N$ -bit integers, compute their product.

**Brute force.**  $N^2$  bit operations.

problem	arithmetic	order of growth
<b>integer multiplication</b>	$a \times b$	$M(N)$
<b>integer division</b>	$a / b, a \bmod b$	$M(N)$
<b>integer square</b>	$a^2$	$M(N)$
<b>integer square root</b>	$\lfloor \sqrt{a} \rfloor$	$M(N)$

integer arithmetic problems with the same complexity as integer multiplication

**Q.** Is brute-force algorithm optimal?

# History of complexity of integer multiplication

---

year	algorithm	order of growth
?	<b>brute force</b>	$N^2$
1962	<b>Karatsuba</b>	$N^{1.585}$
1963	<b>Toom-3, Toom-4</b>	$N^{1.465}$ , $N^{1.404}$
1966	<b>Toom-Cook</b>	$N^{1+\varepsilon}$
1971	<b>Schönhage-Strassen</b>	$N \log N \log \log N$
2007	<b>Fürer</b>	$N \log N 2^{\log^* N}$
?	?	$N$

number of bit operations to multiply two  $N$ -bit integers

used in Maple, Mathematica, gcc, cryptography, ...

**Remark.** GNU Multiple Precision Library uses one of five different algorithm depending on size of operands.





# Numerical linear algebra reductions

**Matrix multiplication.** Given two  $N$ -by- $N$  matrices, compute their product.

**Brute force.**  $N^3$  flops.



# Numerical linear algebra reductions

---

**Matrix multiplication.** Given two  $N$ -by- $N$  matrices, compute their product.

**Brute force.**  $N^3$  flops.

problem	linear algebra	order of growth
<b>matrix multiplication</b>	$A \times B$	$MM(N)$
<b>matrix inversion</b>	$A^{-1}$	$MM(N)$
<b>determinant</b>	$ A $	$MM(N)$
<b>system of linear equations</b>	$Ax = b$	$MM(N)$
<b>LU decomposition</b>	$A = LU$	$MM(N)$
<b>least squares</b>	$\min \ Ax - b\ _2$	$MM(N)$

numerical linear algebra problems with the same complexity as matrix multiplication

**Q.** Is brute-force algorithm optimal?

# History of complexity of matrix multiplication

---

year	algorithm	order of growth
?	<b>brute force</b>	$N^3$
1969	<b>Strassen</b>	$N^{2.808}$
1978	<b>Pan</b>	$N^{2.796}$
1979	<b>Bini</b>	$N^{2.780}$
1981	<b>Schönhage</b>	$N^{2.522}$
1982	<b>Romani</b>	$N^{2.517}$
1982	<b>Coppersmith–Winograd</b>	$N^{2.496}$
1986	<b>Strassen</b>	$N^{2.479}$
1989	<b>Coppersmith–Winograd</b>	$N^{2.376}$
2010	<b>Strother</b>	$N^{2.3737}$
2012	<b>Williams</b>	$N^{2.372873}$
2014	<b>de Gall</b>	$N^{2.372864}$
?	?	$N^{2 + \varepsilon}$

number of floating–point operations to multiply two N–by–N matrices



<http://algs4.cs.princeton.edu>

## 6.5 REDUCTIONS

---

- ▶ *introduction*
- ▶ *designing algorithms*
- ▶ *establishing lower bounds*
- ▶ *classifying problems*
- ▶ *intractability*

# Bird's-eye view

---

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

**Goal.** Prove that a problem is intractable.

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?

input size =  $c + \lg K$



using forced capture rule



*Alan designed the perfect computer*



**Frustrating news.** Very few successes.

# A core problem: satisfiability

---

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

**Ex.**

$$\neg x_1 \quad \text{or} \quad x_2 \quad \text{or} \quad x_3 \quad = \quad \text{true}$$

$$x_1 \quad \text{or} \quad \neg x_2 \quad \text{or} \quad x_3 \quad = \quad \text{true}$$

$$\neg x_1 \quad \text{or} \quad \neg x_2 \quad \text{or} \quad \neg x_3 \quad = \quad \text{true}$$

$$\neg x_1 \quad \text{or} \quad \neg x_2 \quad \text{or} \quad \quad \text{or} \quad x_4 \quad = \quad \text{true}$$

$$\quad \neg x_2 \quad \text{or} \quad x_3 \quad \text{or} \quad x_4 \quad = \quad \text{true}$$

**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.
- Mean field diluted spin glass model in physics.
- Electronic design automation (EDA) for hardware.
- ...

# 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.



Q. Can we do anything substantially more clever?

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

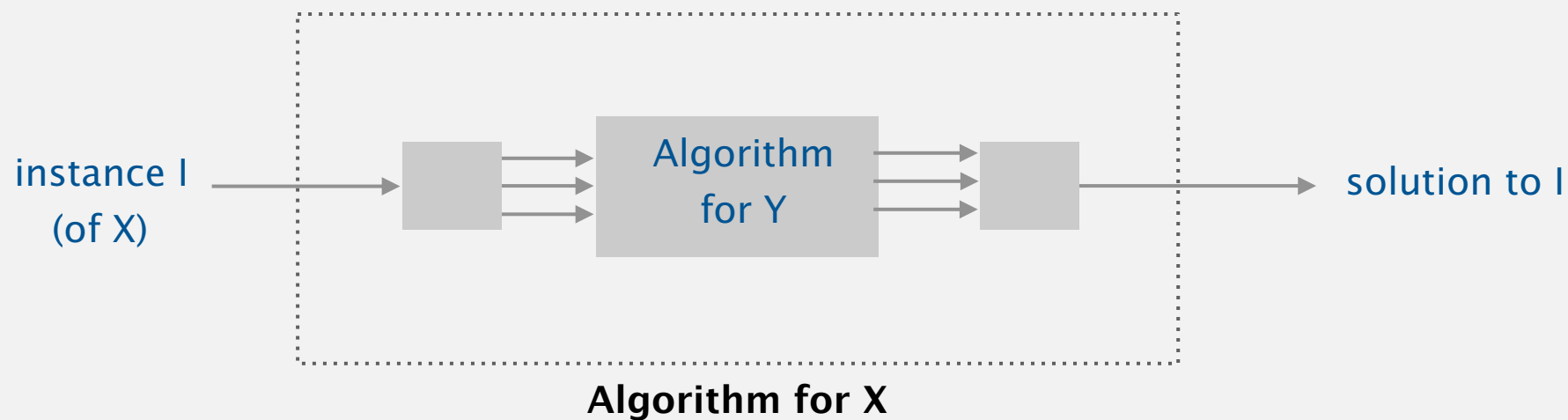
consensus opinion

# Polynomial-time reductions

---

Problem  $X$  **poly-time (Cook) 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.

$$\begin{array}{rcccccccl} \neg x_1 & or & x_2 & or & x_3 & & = & true \\ x_1 & or & \neg x_2 & or & x_3 & & = & true \\ \neg x_1 & or & \neg x_2 & or & \neg x_3 & & = & true \\ \neg x_1 & or & \neg x_2 & or & & or & x_4 & = & true \\ & & \neg x_2 & or & x_3 & or & x_4 & = & true \end{array}$$

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

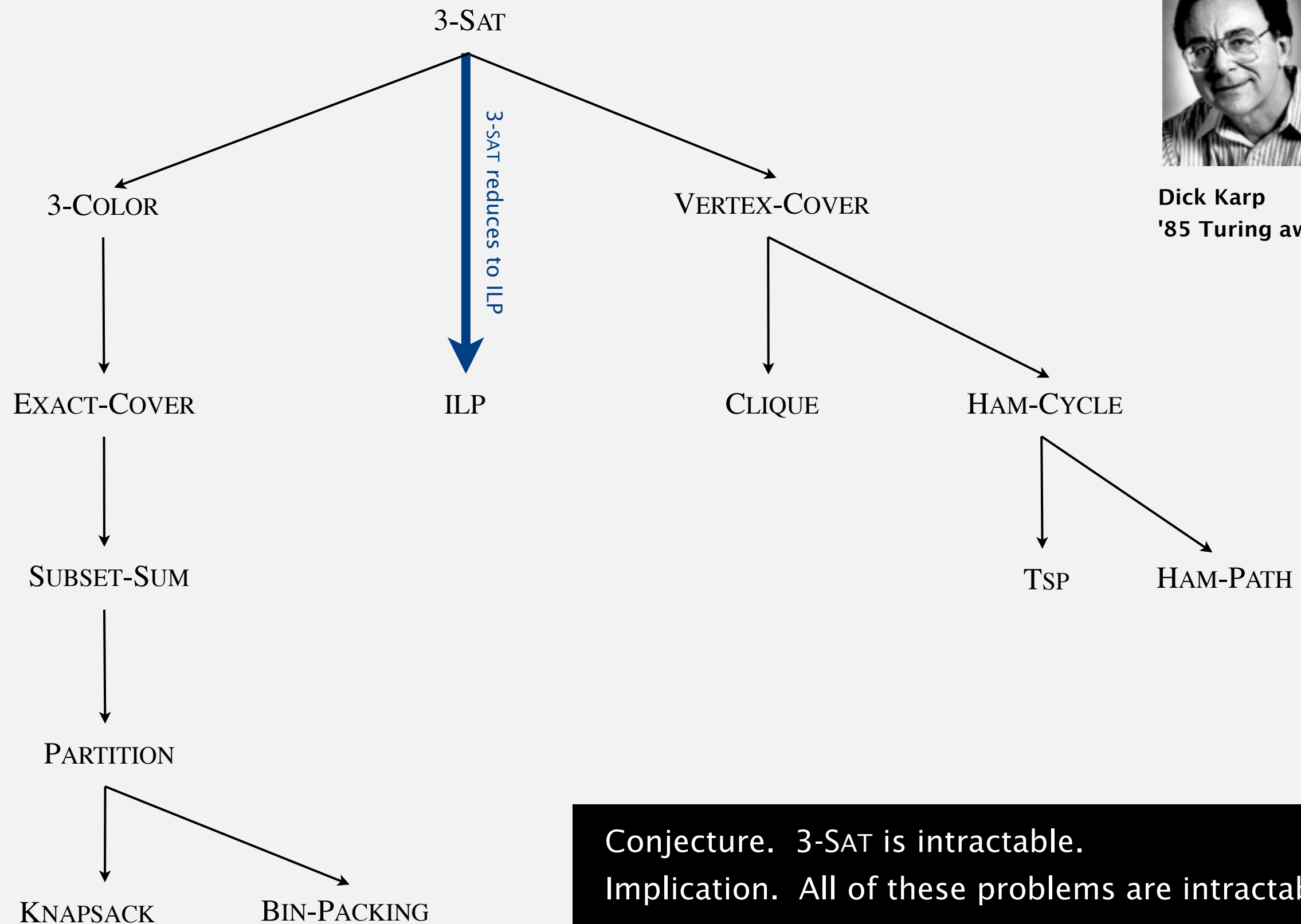
$$\begin{array}{rcccccccl} (1 - x_1) & + & x_2 & + & x_3 & & \geq & 1 \\ x_1 & + & (1 - x_2) & + & x_3 & & \geq & 1 \\ (1 - x_1) & + & (1 - x_2) & + & (1 - x_3) & & \geq & 1 \\ (1 - x_1) & + & (1 - x_2) & + & & + & x_4 & \geq & 1 \\ & & (1 - x_2) & + & x_3 & + & x_4 & \geq & 1 \end{array}$$

**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

---

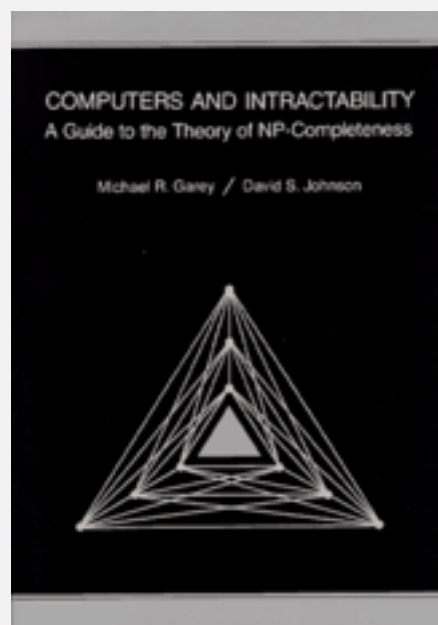
Establishing intractability through poly-time reduction is an important tool in guiding algorithm design efforts.

**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

---

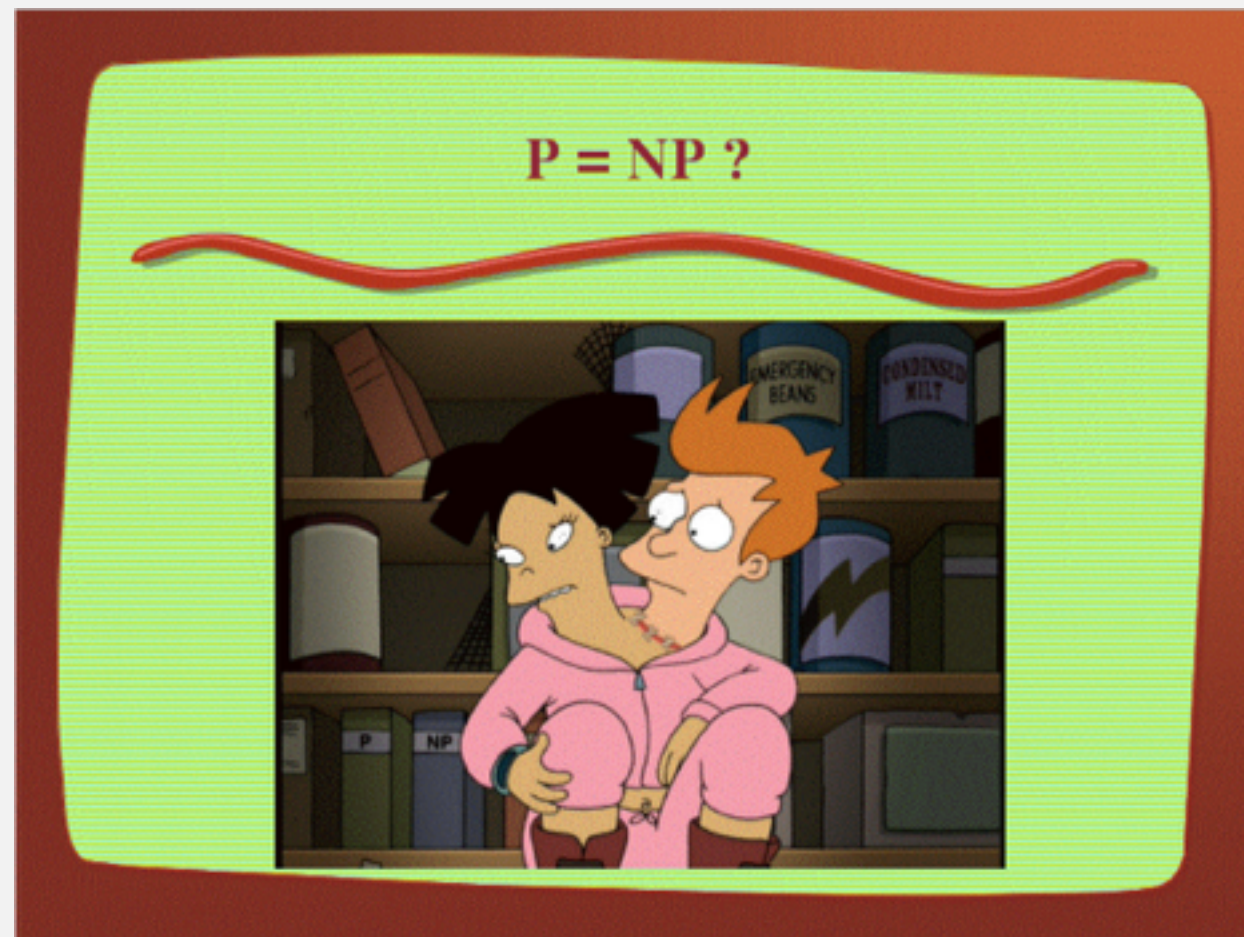
**P.** Set of search problems solvable in poly-time.

**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

---

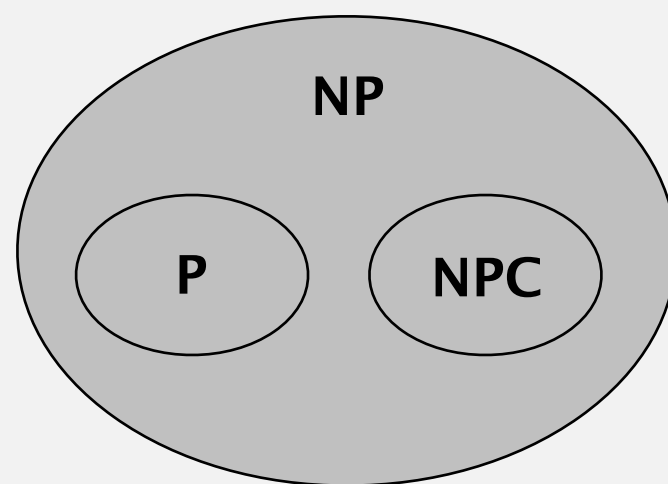
A problem is **NP-COMPLETE** if

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

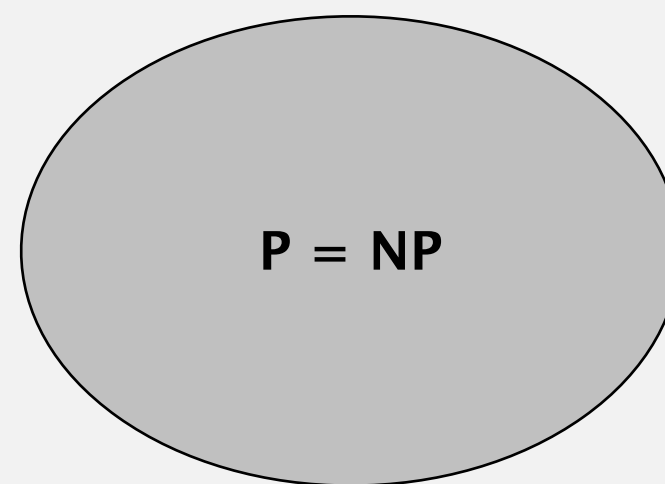
**Cook-Levin theorem.** 3-SAT is **NP-COMPLETE**.

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

Two worlds.

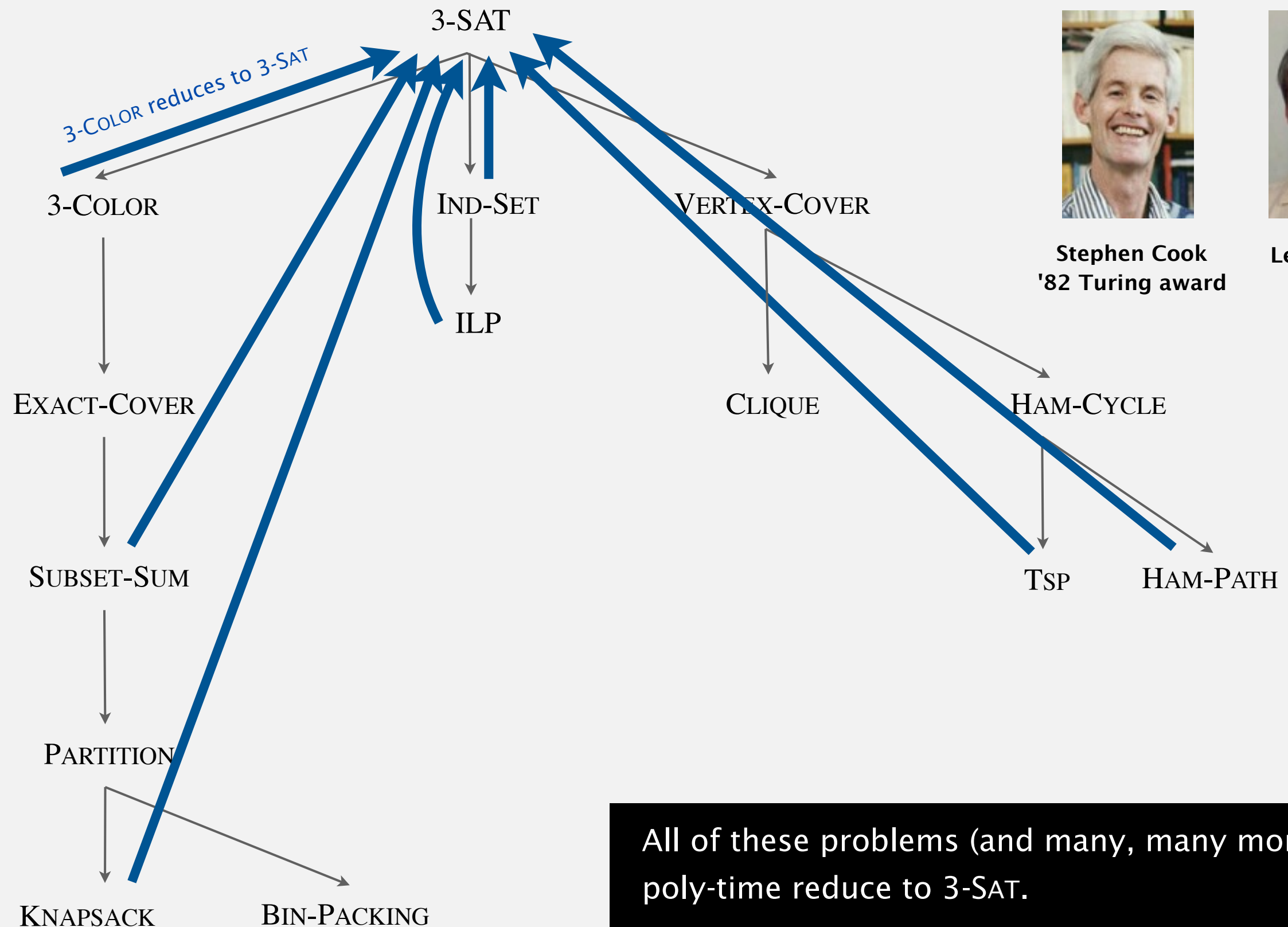


**P ≠ NP**



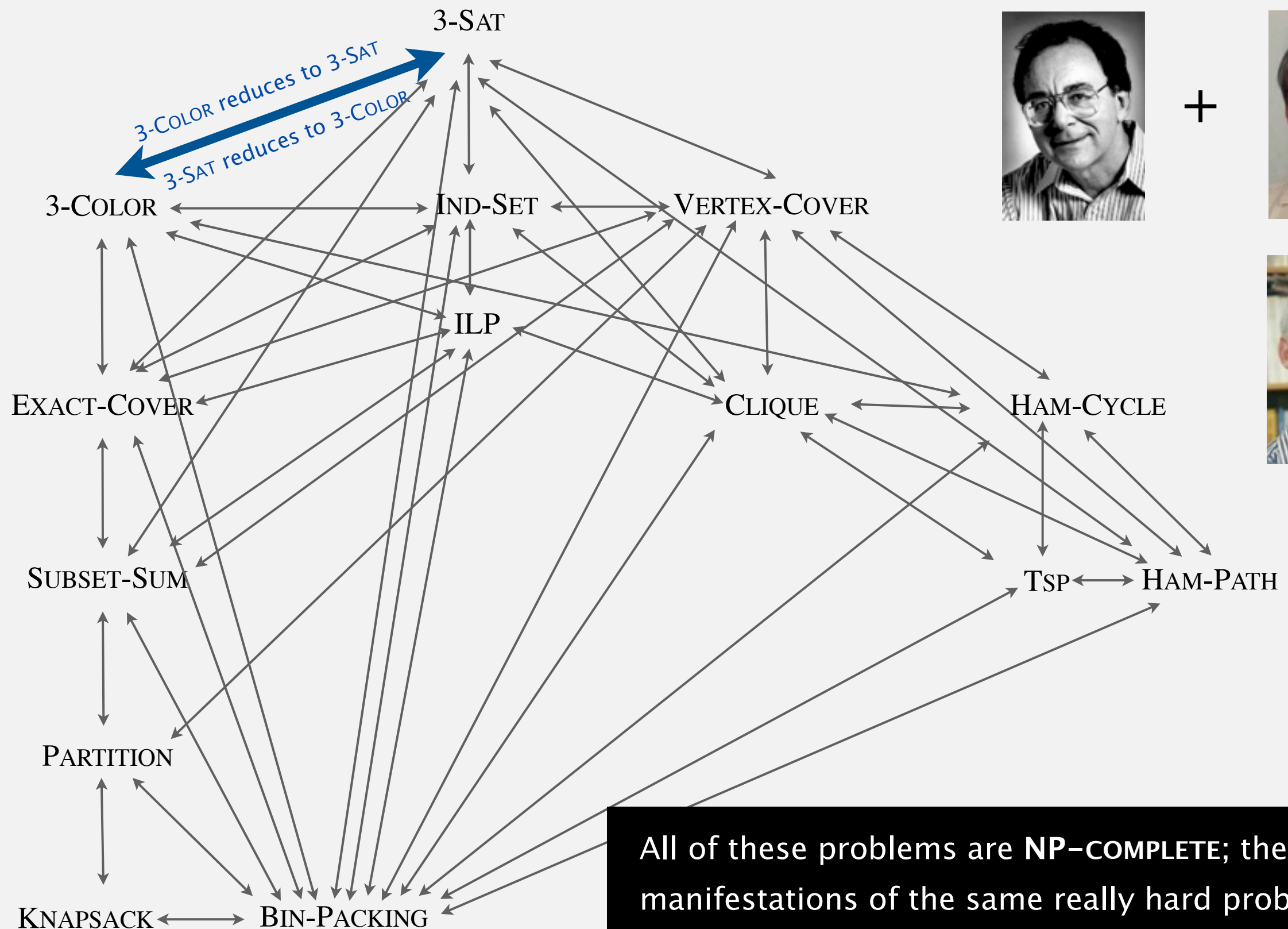
**P = NP**

# Implications of Cook-Levin theorem

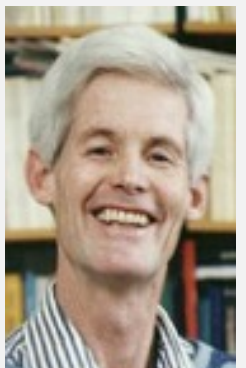




# Implications of Karp + Cook-Levin



+



# Birds-eye view: review

---

**Goal.** Classify **problems** according to computational requirements.

complexity	order of growth	examples
<b>linear</b>	$N$	<i>min, max, median, Burrows-Wheeler transform, ...</i>
<b>linearithmic</b>	$N \log N$	<i>sorting, element distinctness, ...</i>
<b>quadratic</b>	$N^2$	?
$\vdots$	$\vdots$	$\vdots$
<b>exponential</b>	$c^N$	?

**Frustrating news.** Huge number of problems have defied classification.

# Birds-eye view: revised

---

**Goal.** Classify **problems** according to computational requirements.

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

**Good news.** Can put many problems into equivalence classes.

**Complexity class.** Set of problems sharing some computational property.



<https://complexityzoo.uwaterloo.ca>

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

# Summary

---

## Reductions are important in theory to:

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



## 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.

