Suppose...

An alien species is traveling towards Earth and wishes to avoid bloodshed before they arrive.

They want to send a light speed transmission of a proof of their scientific and technological superiority:

- They can only send binary data.
- They do not know our language.

What sequence of bits would prove their superiority?

(Hey, we did warn you that things were going to get weird.)

http://algs4.cs.princeton.edu

BEYOND 226

- *Intro*
- *Reductions*
- *NP completeness*
- *Dealing with NP (or harder) problems*
- *P=NP*
- *Beyond complexity classes*

http://algs4.cs.princeton.edu

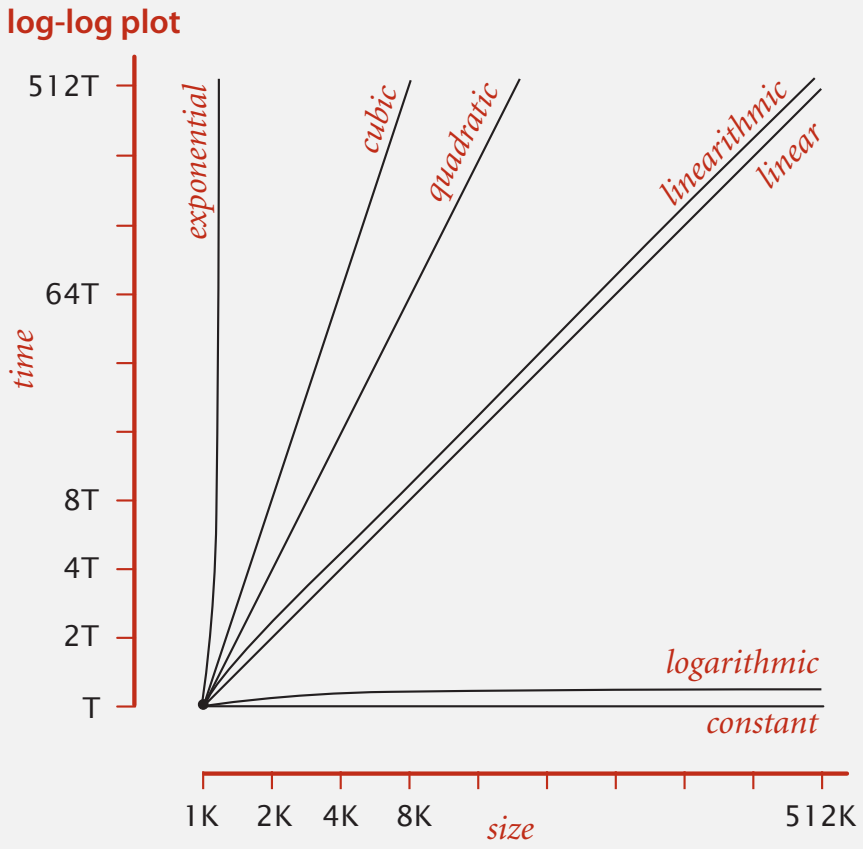
Beyond 226

- *Intro*
- *Reductions*
- *NP completeness*
- *Dealing with NP (or harder) problems*
- *P=NP*
- *Beyond complexity classes*

Overview: introduction to advanced topics

Main topics.

- Most of our problems so far have been easy.
  - Sorting, symbol table operations (array, LLRB, hash table, tries), graph search, MSTs, SPTs, substring matching, regex simulation, etc.
- Some have been hard.
  - 8puzzle.
  - Hamilton path.



Typical orders of growth

What is easy?

Polynomial Time Solvability

- A problem is in P if there is an algorithm that solves it in $O(N^k)$ time.
  - $O(N^k)$ - Worst case order of growth is $\leq N^k$.
  - N is number of bits needed to specify input.

(Technically speaking, the problem is in P only if it is a yes/no problem, but we'll ignore this)

Example

- Sorting Q items of c bits each using `compareTo()` as our model of computation.
  - Total bits used: $N = cQ$
  - Mergesort, worst case order of growth: $Q \log Q$
  - In terms of big O: $O(Q^2) = O(N^2/c^2) = O(N^2)$

What is easy?

Polynomial Time Solvability

- A problem is in P if there is an algorithm that solves it in $O(N^k)$ time.
  - $O(N^k)$ - Worst case order of growth is $\leq N^k$.
  - N is number of bits needed to specify input.

Example

- Sorting Q random items of W characters of c bits each using `charAt()` as our model of computation.
  - Total bits used: N = cWQ
  - Mergesort, worst case order of growth: WQ log Q
  - In terms of big O, $O(WQ^2)$
  - Fastest growth if bits are used for W or Q or some combination?

What is easy?

Polynomial Time Solvability

- A problem is in P if there is an algorithm that solves it in $O(N^k)$ time.
  - $O(N^k)$ - Worst case order of growth is $\leq N^k$.
  - N is number of bits needed to specify input.

Example

- Sorting Q random items of W characters of c bits each using `charAt()` as our model of computation.
  - Total bits used: N = cWQ
  - Mergesort, worst case order of growth: WQ log Q
  - In terms of big O, $O(WQ^2)$
  - Fastest growth if bits are used for W or Q or some combination? Q

What is easy?

Polynomial Time Solvability

- A problem is in P if there is an algorithm that solves it in $O(N^k)$ time.
  - $O(N^k)$ - Worst case order of growth is $\leq N^k$.
  - N is number of bits needed to specify input.

Example

- Sorting Q random items of W characters of c bits each using `charAt()` as our model of computation.
  - Total bits used: N = cWQ
  - Mergesort, worst case order of growth: WQ log Q
  - In terms of big O, $O(WQ^2)$
  - Fastest growth if bits are used for W or Q or some combination? Q
  - $O(WQ^2) = O(N^2/c^2) = O(N^2)$

P

Polynomial Time Solvability

- A problem is in P if there is an algorithm that solves it in $O(N^k)$ time.
  - Worst case order of growth is $\leq N^k$.
  - N is number of bits needed to specify input.

	Order of Growth	Input bits	
Finding Maximum	Q	$Q \propto N$	$O(N)$
Sorting with compareTo	Q log Q	$Q \propto N$	$O(N^2)$
Sort: Compare charAt()	QW log Q	$Q \propto N$	$O(N^2)$
DFS and BFS	E + V	$V, E \propto N$	$O(N)$
Baseball Elimination	T^6	$T \propto N^2$	$O(N^3)$

Easy as P

Why $O(N^k)$?

- P seems rather generous.
- $O(N^k)$ closed under addition and multiplication.
  - Consecutively run two algorithms in P, still in P.
  - Run an algorithm N times, still in P.
- Exponents for practical problems are typically small.

A modern standard for simplicity

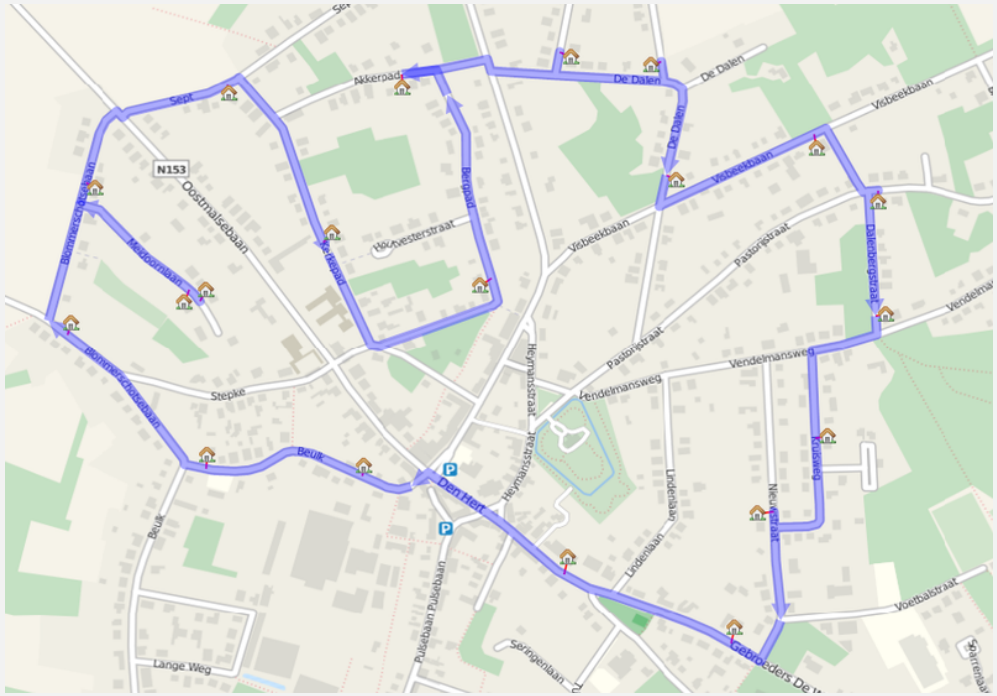
Most important point

- If a practical problem is easy, it is in P.
- If a practical problem is in P, it is easy.

Not everything is easy

Difficult Problems

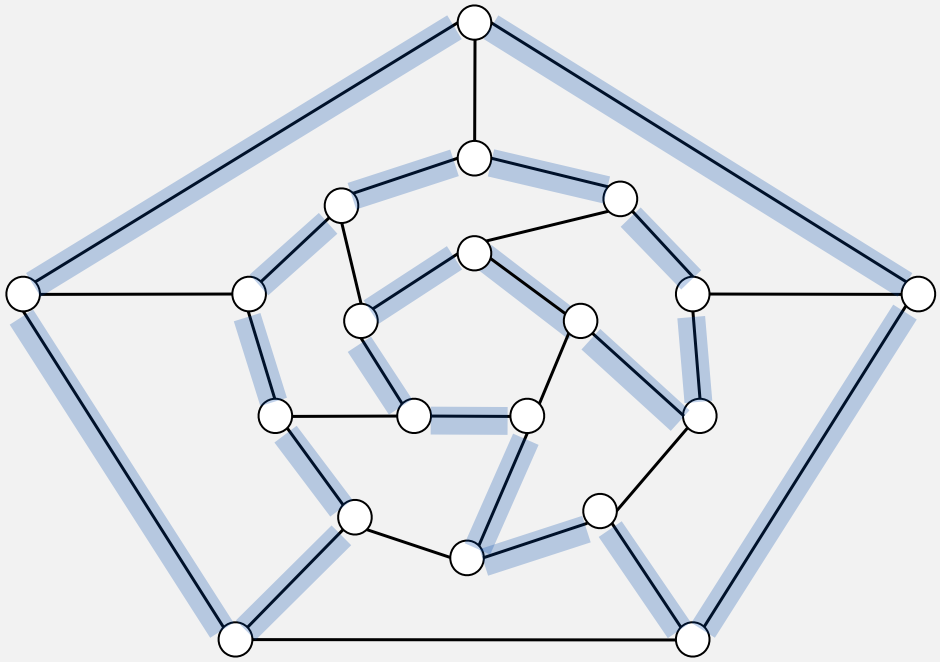
- TSPM (Traveling Salesperson Problem with Multiple Visits)
  - Given weighted directed graph.
  - Finds tour of vertices (vertices may be used multiple times) with minimum weight. Arrives back at start.
  - Conjectured to be outside P.
- More general that it seems!

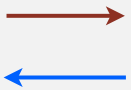


Challenge

Difficult Problems

- TSPM (Traveling Salesperson Problem with Multiple Visits).
  - Finds tour of vertices (vertices may be used multiple times) with minimum weight. Arrives back at start.
- Goal
  - Use TSPM to solve the undirected Hamilton Cycle problem.





???

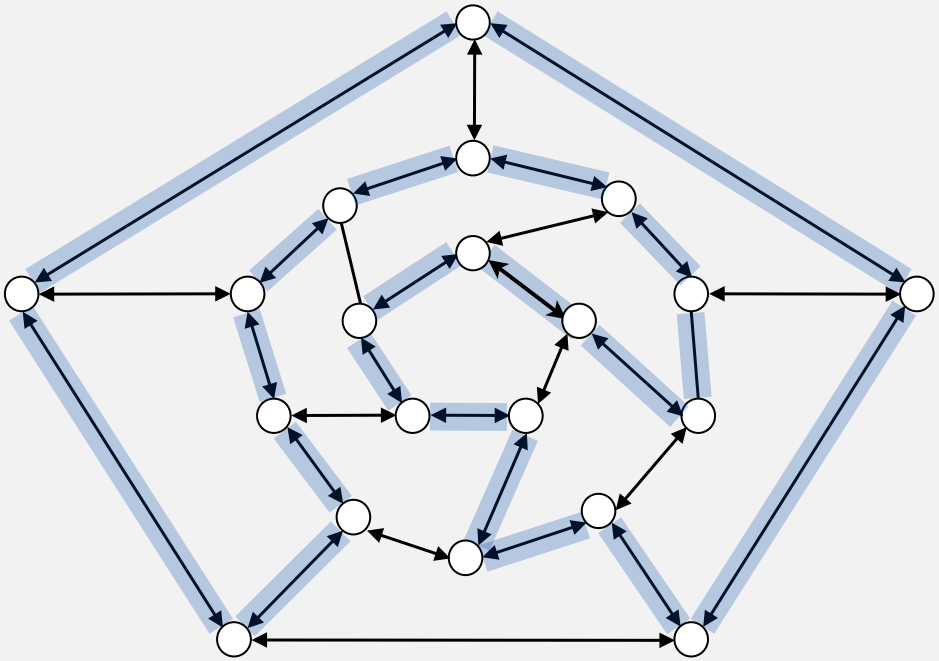
Challenge

Difficult Problems

- TSPM (Traveling Salesperson Problem with Multiple Visits)
  - Finds tour of vertices (vertices may be used multiple times) with minimum weight. Arrives back at start.
- Goal
  - Use TSPM to solve the Hamilton Cycle problem.

Solution

- Create directed graph with all edge weights 1.
  - Two directed edges for each original undirected edge.
- Run TSPM on graph.
- If total weight of tour is |V| then we have a Hamilton Cycle.



TSP

TSP is an incredibly powerful tool

- Nearly any computational problem you care to solve.
  - Protein folding.
  - Sudoku.
  - Proving mathematical theorems.

TSP is probably (?) very hard

- Edmonds *Paths, Trees, and Flowers* (1965):
  - There is no *good* [polynomial time] solution to TSP.
- Many interesting problems reduce to TSP [TSP solves MIPs].
- People have been trying to solve TSP for 50 years.

TSP can provide proof that your problem is hard

- Suppose TSP reduces to your problem of interest. [Your problem solves TSP]
  - Your problem is probably not efficiently solvable.
- Other techniques to come!

Problem

Difficult problems

- Many practical problems are unsolvable using the tools of COS226.
  - Most of these problems are actually TSP in disguise (!!).

- Learning to recognize TSP equivalent problems (NP Complete).
  - Need some rigorous notion of equivalent difficulty.
  - Need some rigorous notion of TSP's difficulty.

http://algs4.cs.princeton.edu

Beyond 226

- *Intro*
- *Reductions*
- *NP completeness*
- *Dealing with NP (or harder) problems*
- *P=NP*
- *Beyond complexity classes*

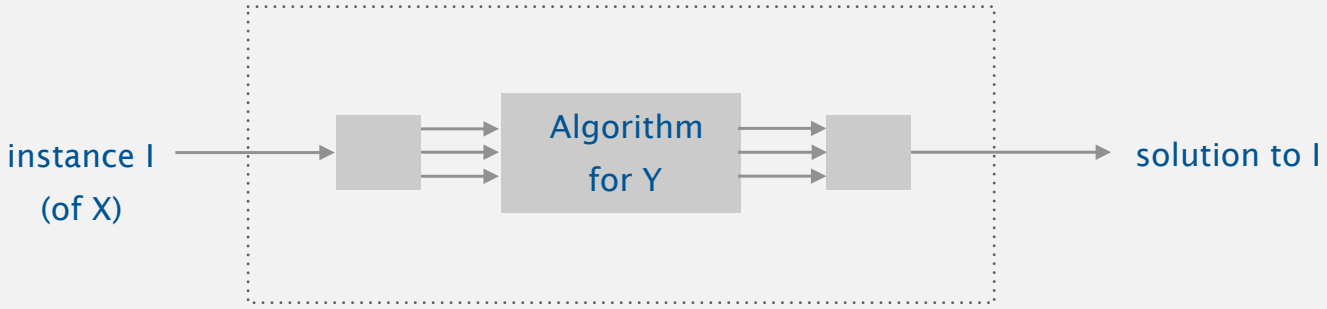
Examples of reduction

- Sudoku reduces to TSP

- Your MAT 213 homework reduces to TSP
- Some of it reduces to even easier problems

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.

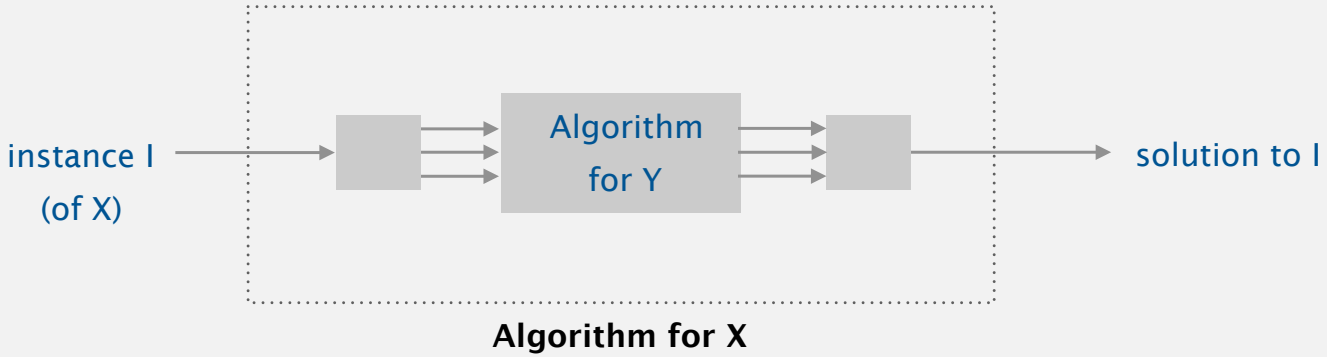
perhaps many calls to Y
on problems of different sizes
(though, typically only one call)

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

X is no harder than Y (same or lesser difficulty).

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.

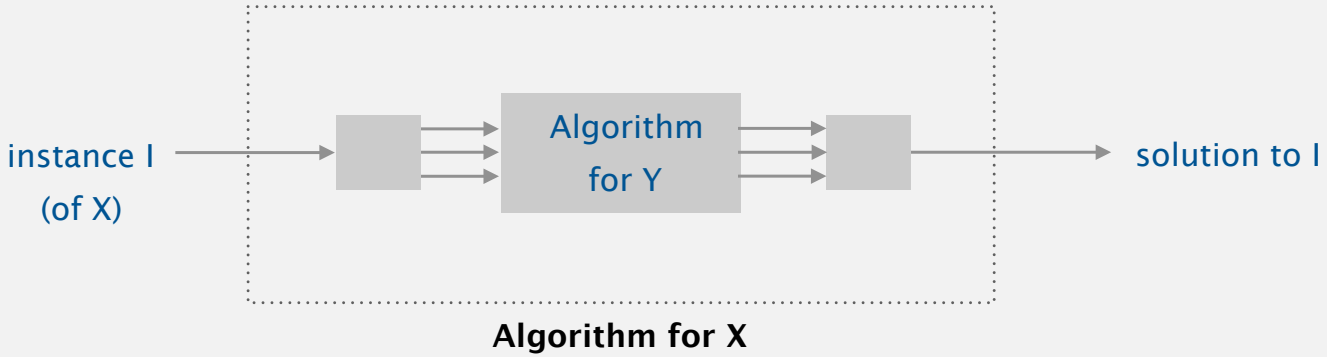
Cost of solving finding the median. $N \log N + 1$.

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.

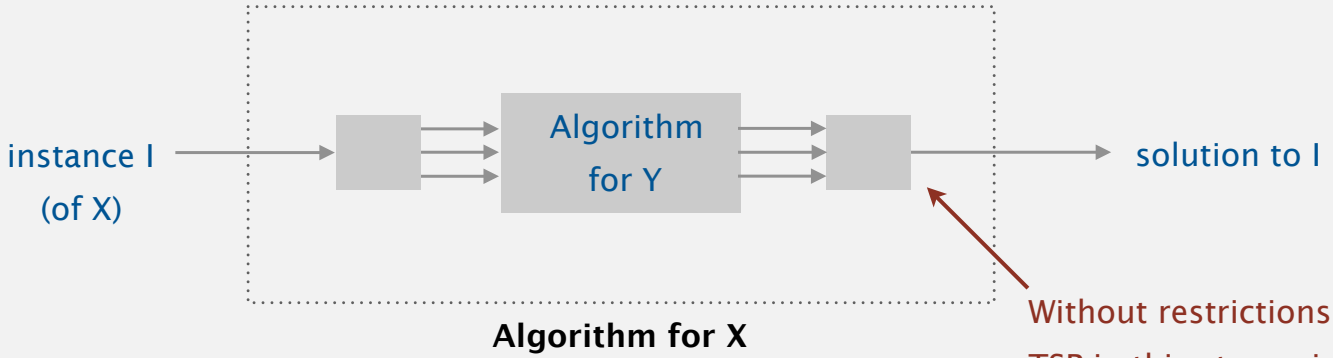
cost of sorting

cost of reduction

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

Reduction

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



Ex 3. [TSP reduces to sorting]

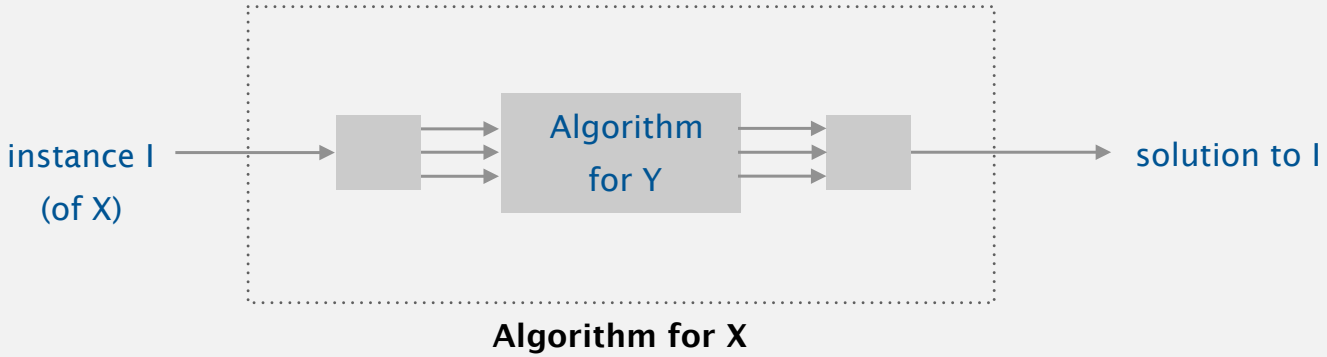
Use Brute force.

Definition didn't say anything about how costly the reduction can be!

(Oops.)

Reduction

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

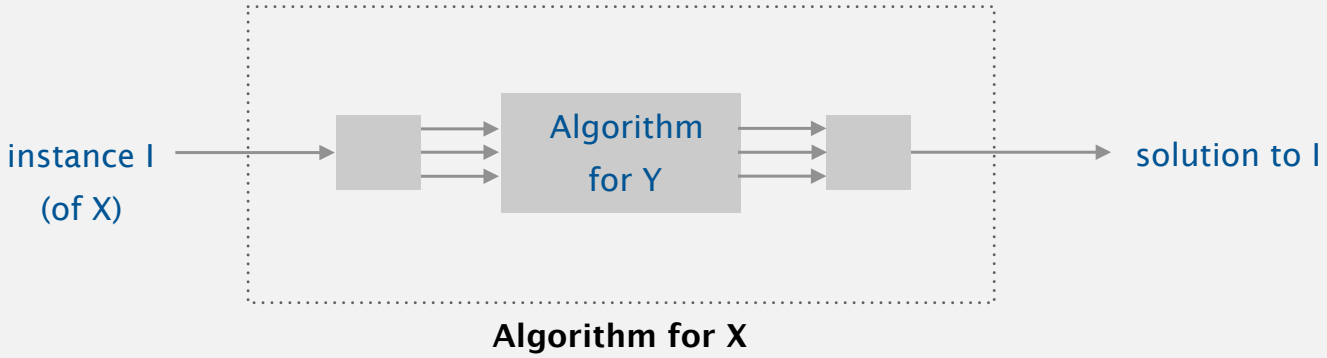


Ex 4. Sorting reduces to TSP.

Don't even use the algorithm for TSP, just do sorting directly.

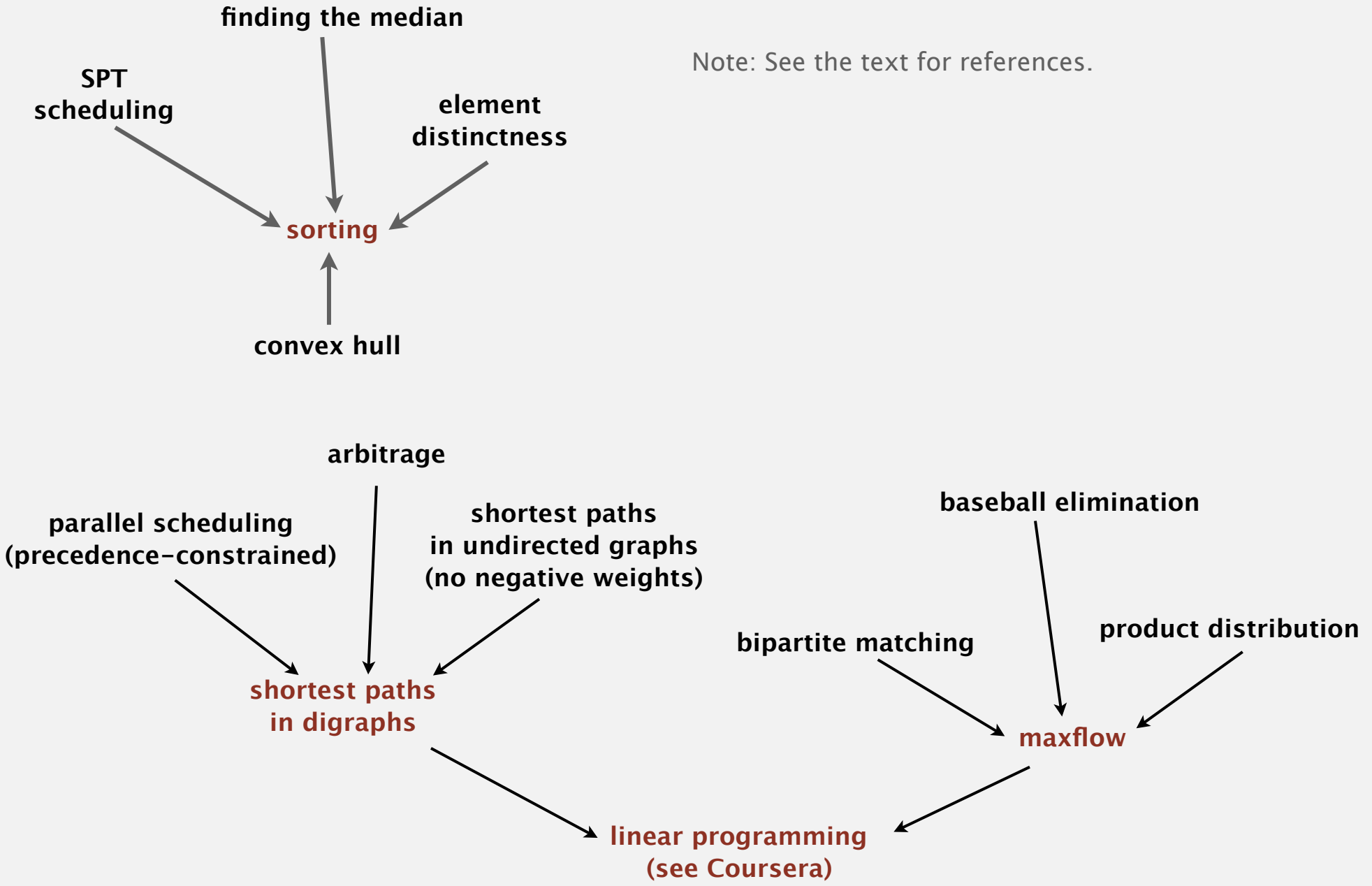
Reduction

Def. Problem X linear-time reduces to problem Y if X reduces to Y with linear reduction cost and constant number of calls to Y.



Also common: polynomial-time reduction.

Linear-time reductions involving familiar problems

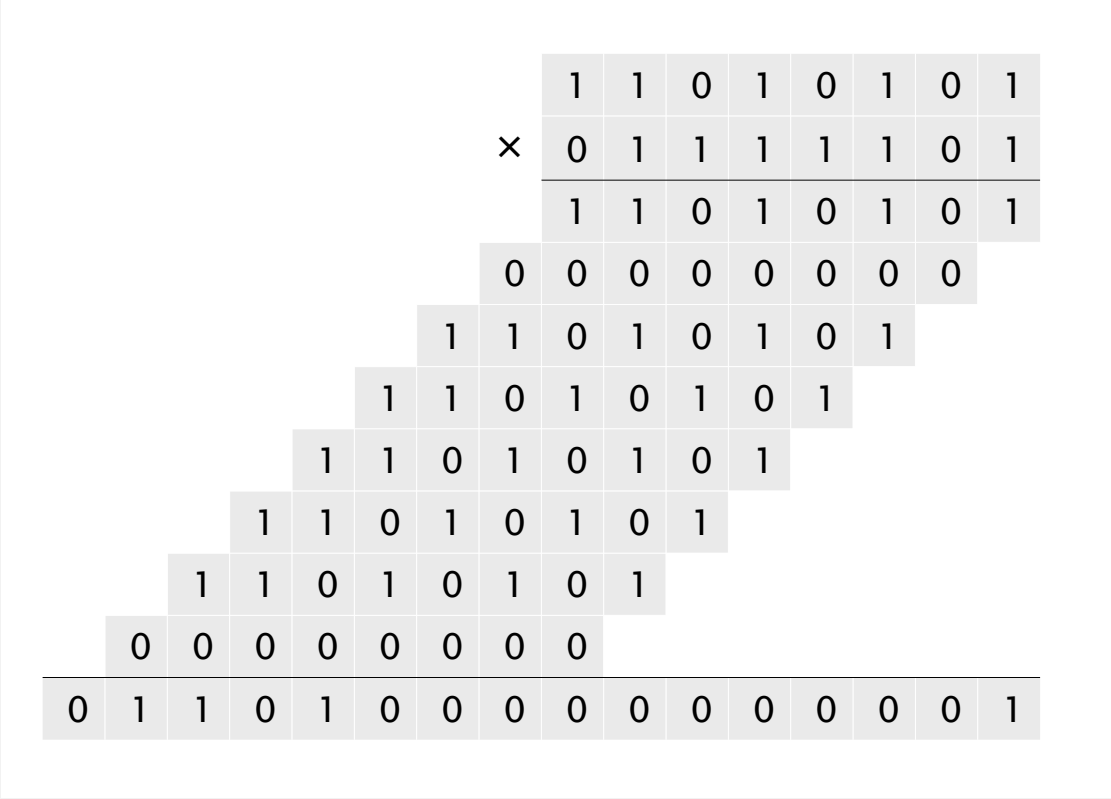


Note: See the text for references.

Integer arithmetic reductions

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

Brute force. N^2 bit operations.



Integer arithmetic reductions

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

Brute force. N^2 bit operations.

problem	arithmetic	order of growth
integer multiplication	a × b	M(N)
integer division	a / b, a mod b	M(N)
integer square	a^2	M(N)
integer square root	$\lfloor \sqrt{a} \rfloor$	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
?	brute force	N^2
1962	Karatsuba	$N^{1.585}$
1963	Toom-3, Toom-4	$N^{1.465}$, $N^{1.404}$
1966	Toom-Cook	$N^{1+\varepsilon}$
1971	Schönhage–Strassen	N log N log log N
2007	Fürer	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.

GMP
«Arithmetic without limitations»

Linear algebra reductions

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

Brute force. N^3 flops.

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
matrix multiplication	$A \times B$	MM(N)
matrix inversion	A^{-1}	MM(N)
determinant	$\lvert A \rvert$	MM(N)
system of linear equations	$Ax = b$	MM(N)
LU decomposition	$A = L\,U$	MM(N)
least squares	$\min \lVert Ax - b \rVert_2$	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
?	brute force	N^3
1969	Strassen	$N^{2.808}$
1978	Pan	$N^{2.796}$
1979	Bini	$N^{2.780}$
1981	Schönhage	$N^{2.522}$
1982	Romani	$N^{2.517}$
1982	Coppersmith-Winograd	$N^{2.496}$
1986	Strassen	$N^{2.479}$
1989	Coppersmith-Winograd	$N^{2.376}$
2010	Strother	$N^{2.3737}$
2011	Williams	$N^{2.3727}$
?	?	$N^{2+\varepsilon}$

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

http://algs4.cs.princeton.edu

Beyond 226

- *Intro*
- *Reductions*
- *NP Completeness*
- *Dealing with NP (or harder) problems*
- *P=NP*
- *Beyond complexity classes*

Decision Problems vs. Function Problems

Decision Problem

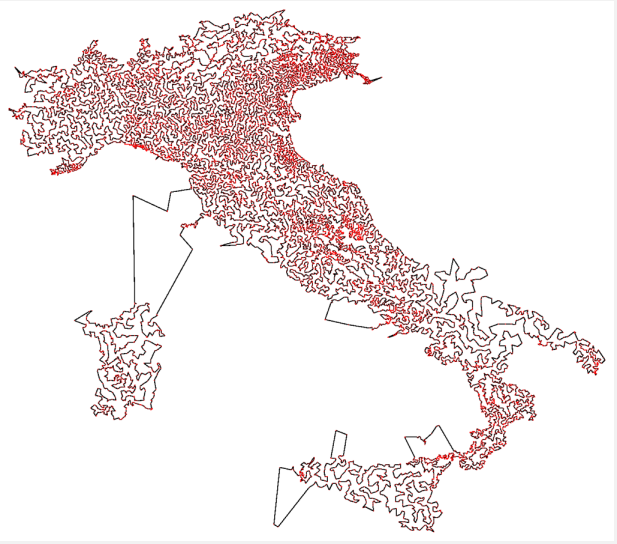
- Given some input, gives “yes” or “no” as answer.

Function problem

- Given some input, give some output as an answer.

Examples:

- Decision problems
  - Does a TSP tour exist of length < M?
  - Is N the product of two primes?
- Function problems
  - What is the minimal weight TSP tour?
  - What are the factors of N?



TSP Tour of Italy’s Cities

NP

The Class NP

- Decision problem.
- If answer is "Yes", a proof exists that can be verified in polynomial time.
  - NP: Does a TSP tour exist of length less than 1000?
  - Not NP: Is a given TSP tour optimal?

- Stands for "non-deterministic polynomial"
  - Name is a confusing relic. Don't worry about it.

- **Most important detail: Verifiable in Polynomial Time.**
  - "In an ideal world it would be renamed P vs VP" - Clyde Kruskal

"Joseph Kruskal [inventor of Kruskal's algorithm] should not be confused with his two brothers Martin Kruskal(1925–2006; co-inventor of solitons and of surreal numbers) and William Kruskal(1919–2005; developed the Kruskal-Wallis one-way analysis of variance), or his nephew Clyde Kruskal." -Dbenbenn

http://en.wikipedia.org/wiki/Joseph_Kruskal

NP

Decision problems are actually useful

- Example: Does there exist a tour of length less than `bestKnownTour`?
- Better example (solves function version of TSP with integer weights):
  - Does there exist a tour of length less than 10000?
  - Less than 5000?
  - Less than 7500?
  - Less than 6250?
  - Repeat until Δ less than 1.

NP

A vast number of interesting well-defined problems are in NP.

- Hand-wavy reason: In NP if you can ask useful decision sub-problems about a solution.
- Example:
  - Does simulated folded protein have energy below X?
  - Is total cost of deploying resources below X?
  - Is array in sorted order? (problem is also in P!)
- Counter-example?
  - Is move X better than move Y in this chess game on N^2 board?

Completeness (short detour)

Completeness

- Let Q be a class of problems and let π be a specific problem.

- π is Q-Complete if
  - π is in Q.
  - Everything in Q reduces to π [π solves any problem in Q].

- If a solution is known, can use π as a tool to solve any problem in Q.

NP-complete

NP-complete

- A problem π is NP-complete if:
  - π is in NP.
  - All problems in NP reduce to π.

- Solution to an NP-complete problem would be a key to the universe!

Two questions

- Are there any NP-complete problems?
- Do we know how to solve any of them?

Existence of an NP complete problem

Also in NP!

3SAT

- Cook (71) and Levin (73) proved that every NP problem reduces to 3SAT.
  - 3SAT is at least as hard as every other problem in NP.
  - A solution to 3SAT provides a solution to every problem in NP.

- Does there exist a truth value for boolean variables that obeys a set of 3-variable disjunctive constraints: (x1 || x2 || !x3) && (x1 || !x1 || x1)

Stephen Cook

Leonid Levin

Existence of an NP complete problem

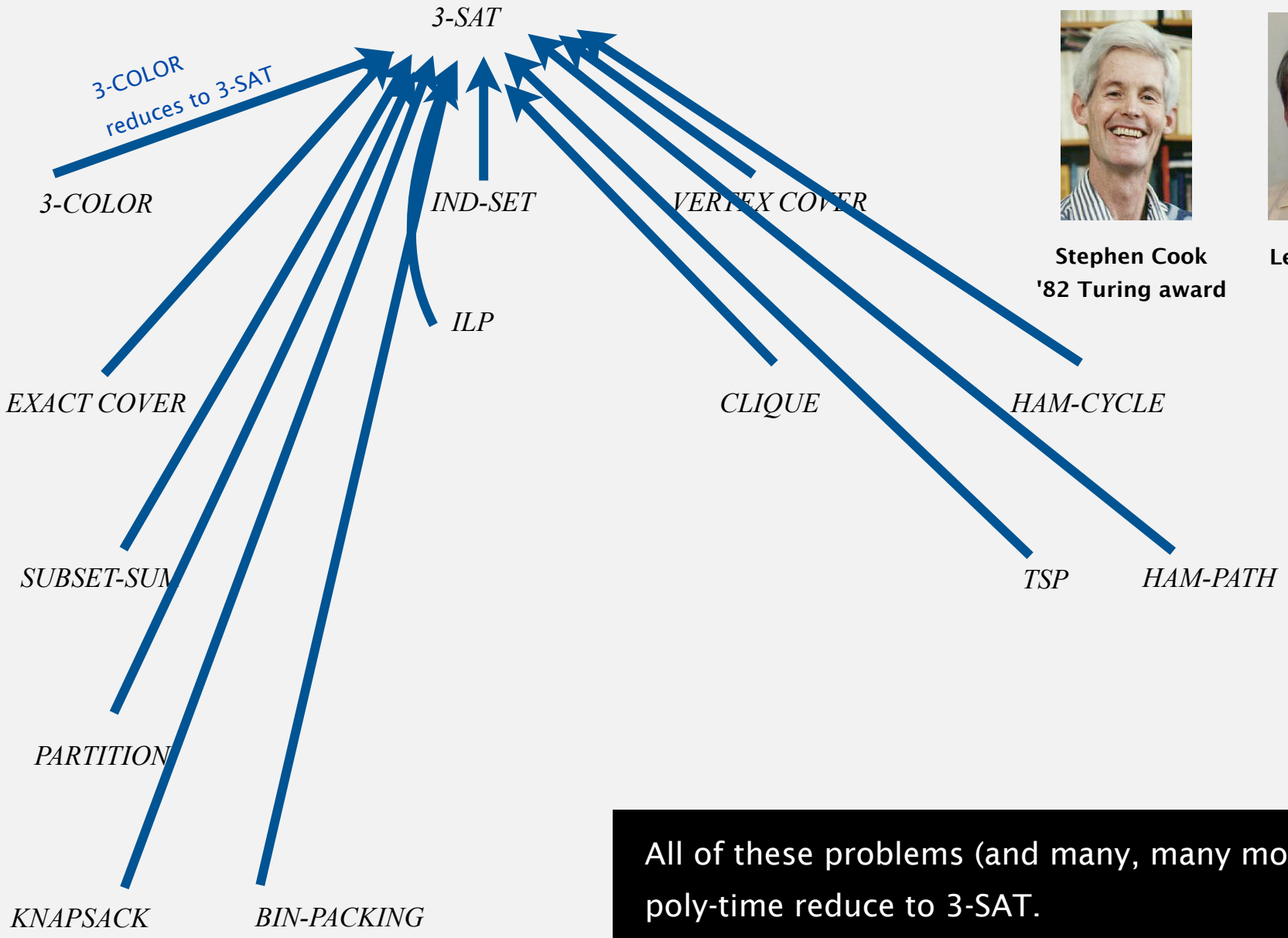
Rough idea of Cook-Levin theorem

- Create giant (!!) boolean logic expression that represents entire state of your computer at every time step.
- If solution takes polynomial time, boolean logic circuit is polynomial in size.
- Example boolean logic variable: True if 57173th bit of memory is true and we're on line 38 of code during cycle 7591872 of execution.

Stephen Cook

Leonid Levin

Implications of Cook-Levin theorem



Stephen Cook
'82 Turing award

Leonid Levin

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

3SAT

Great, 3SAT solves most well defined problems of general interest!

Can we solve 3SAT efficiently?

- Nobody knows how to solve 3SAT efficiently.
- Nobody knows if an efficient solution exists.
  - Unknown if 3SAT is in P.

Other NP Complete problems?

- Are there other keys to this magic kingdom?

NP Complete

There are more

- Dick Karp (72) proved that 3SAT reduces to 21 important NP problems.
  - Example: A solution to TSP provides a solution to 3SAT.
  - All of these problems join 3SAT in the NP Complete club.
- Proof applies only to these 21 problems. Each was its own special case.

Dick Karp

More poly-time reductions from 3-satisfiability

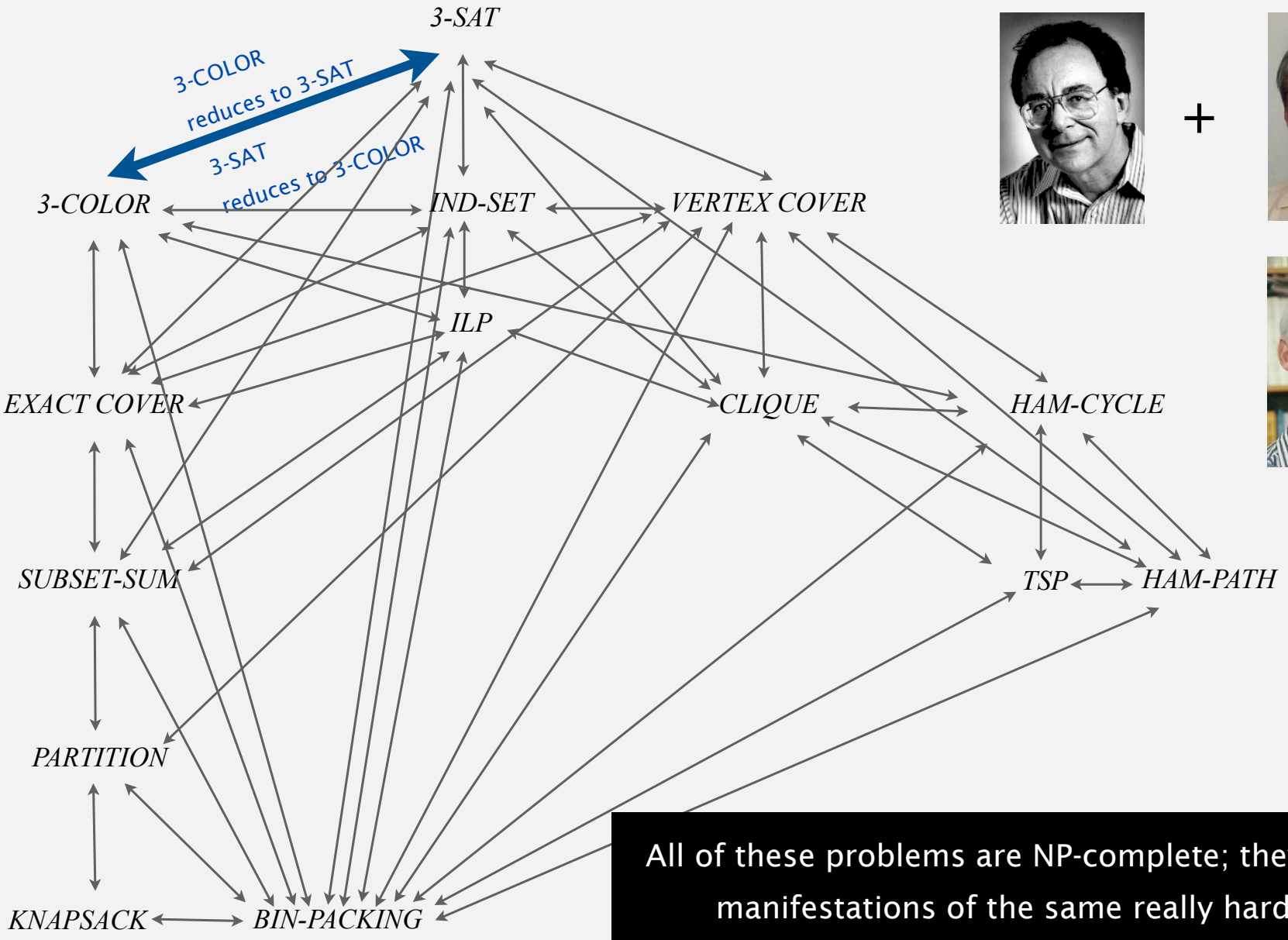
Dick Karp

'85 Turing award

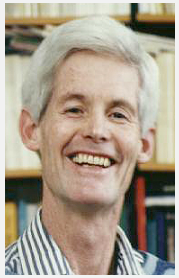
Conjecture. 3-SAT is intractable.

Implication. All of these problems are intractable.

Implications of Karp + Cook-Levin



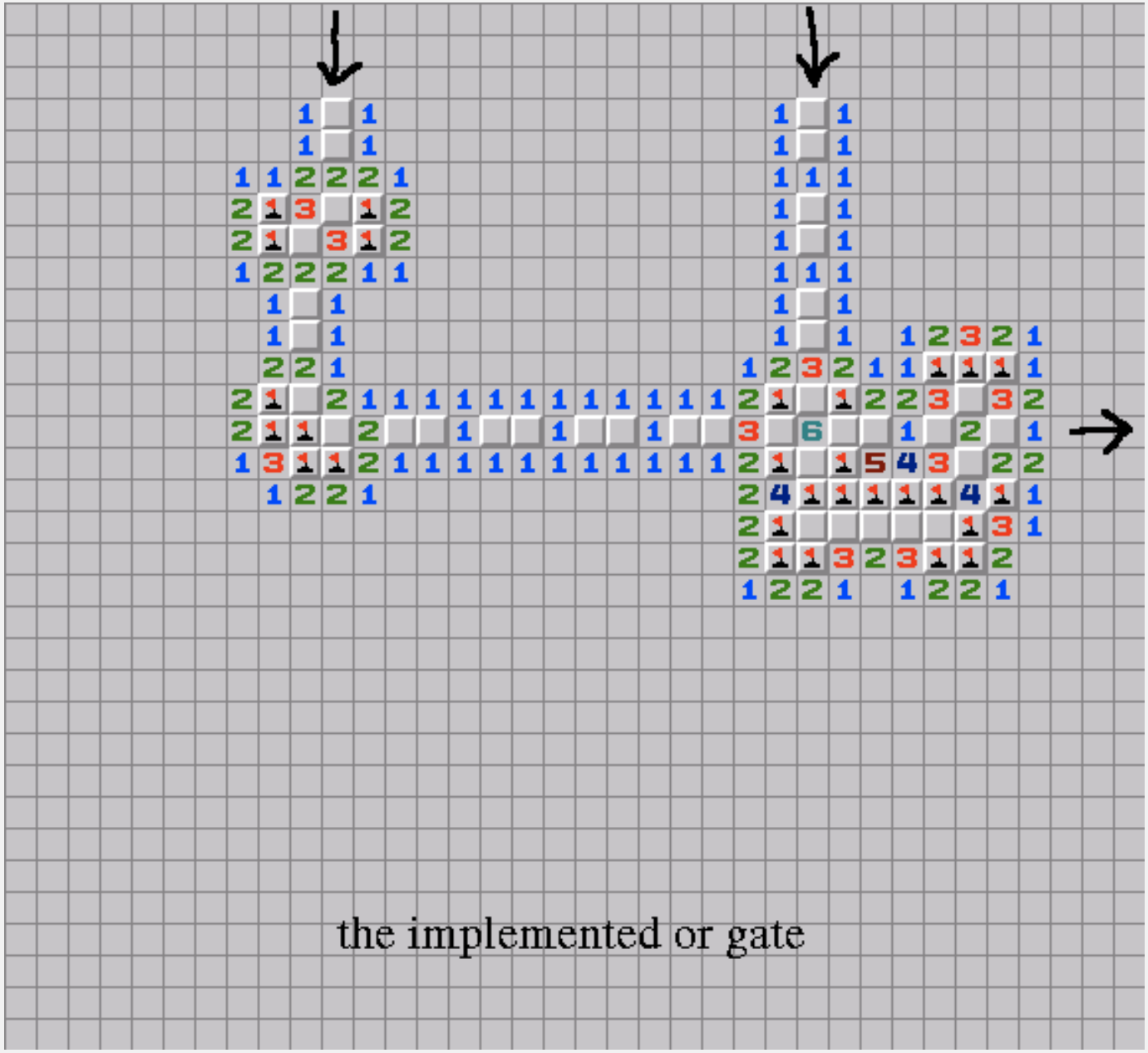
+



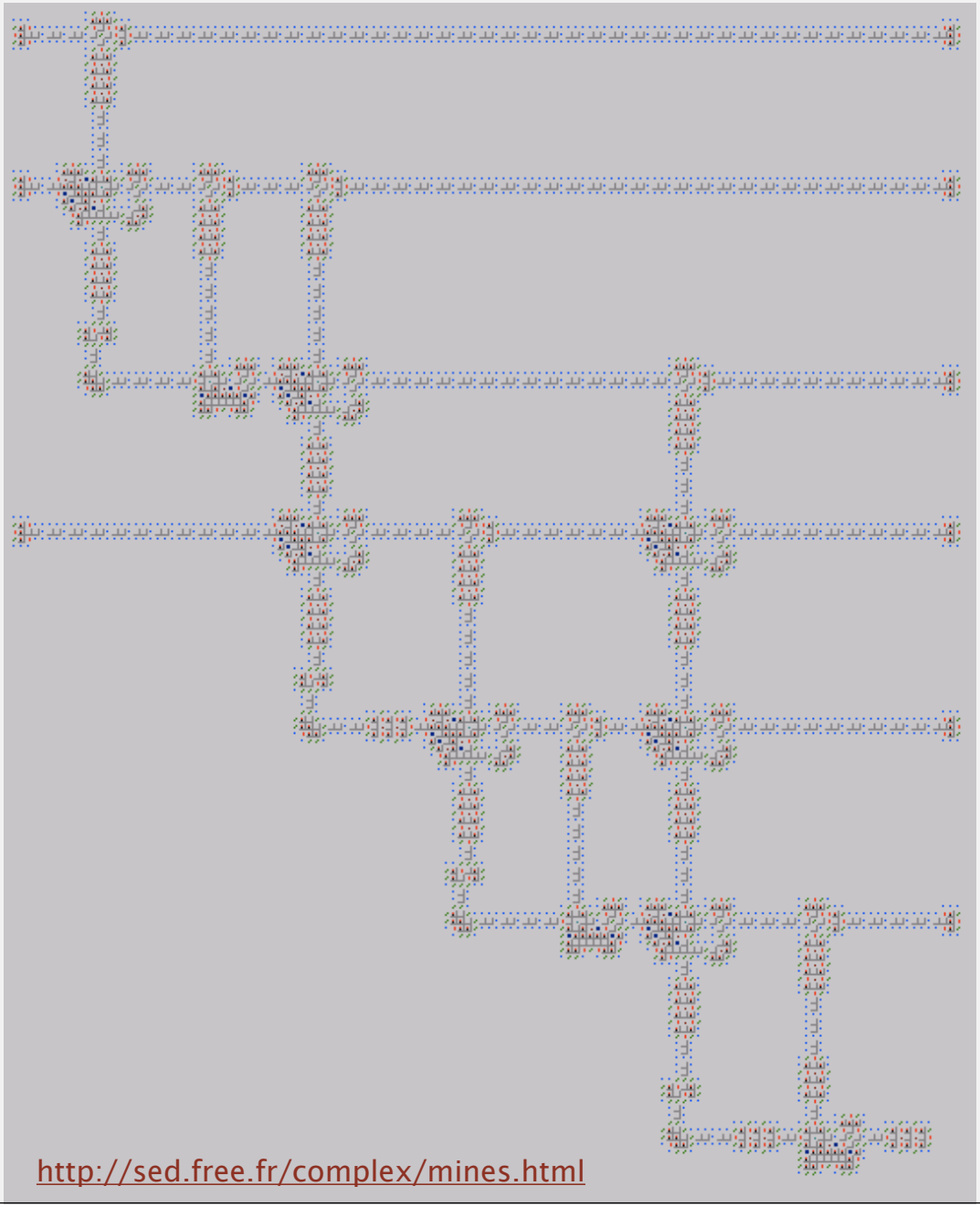
All of these problems are NP-complete; they are manifestations of the same really hard problem.

A familiar NP Complete problem

http://sed.free.fr/complex/mines.html



A familiar NP Complete problem



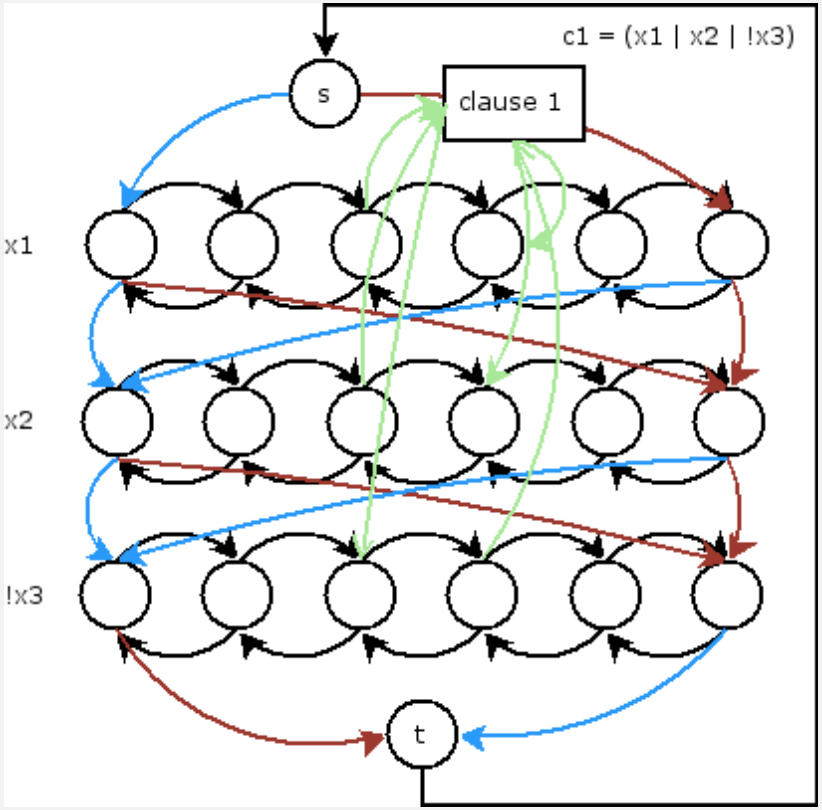
http://sed.free.fr/complex/mines.html

How to tell if your problem is NP Complete?

- Prove that it is in NP [easy].

- Prove that **some** NP Complete problem reduces to your problem [tricky!]

Example of a tricky reduction: SAT to Hamiltonian cycle



http://cs482.elliottback.com/lecture-25-hamiltonian-cycle-problem/

NP completeness in nature

- Protein folding is NP Complete
  - We want to understand structure.
  - Protein crystallography is extremely expensive and slow.
  - Wrong answer may result in dramatically different structure.

- Preview of later: Nature folds proteins
  - Is nature solving an NP complete problem efficiently?

Approximation is usually OK

Most of the time, it's not "all or nothing"

- Bin packing

- Actual traveling salesperson

http://algs4.cs.princeton.edu

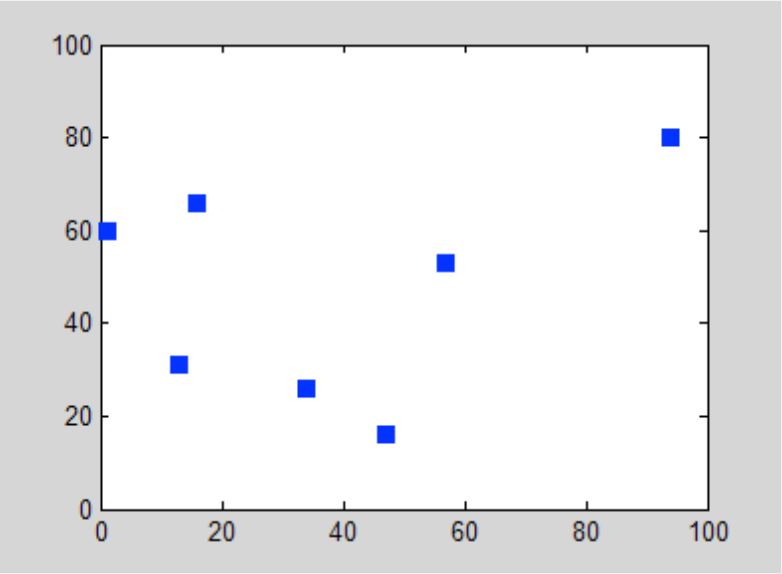
Beyond 226

- *Intro*
- *Reductions*
- *NP Completeness*
- *Dealing with NP (or harder) problems*
- *P=NP*
- *Beyond complexity classes*

Approximation

Approach 1: Approximate Heuristics

- Accept incorrect answers
  - TSP, always go to closest city next

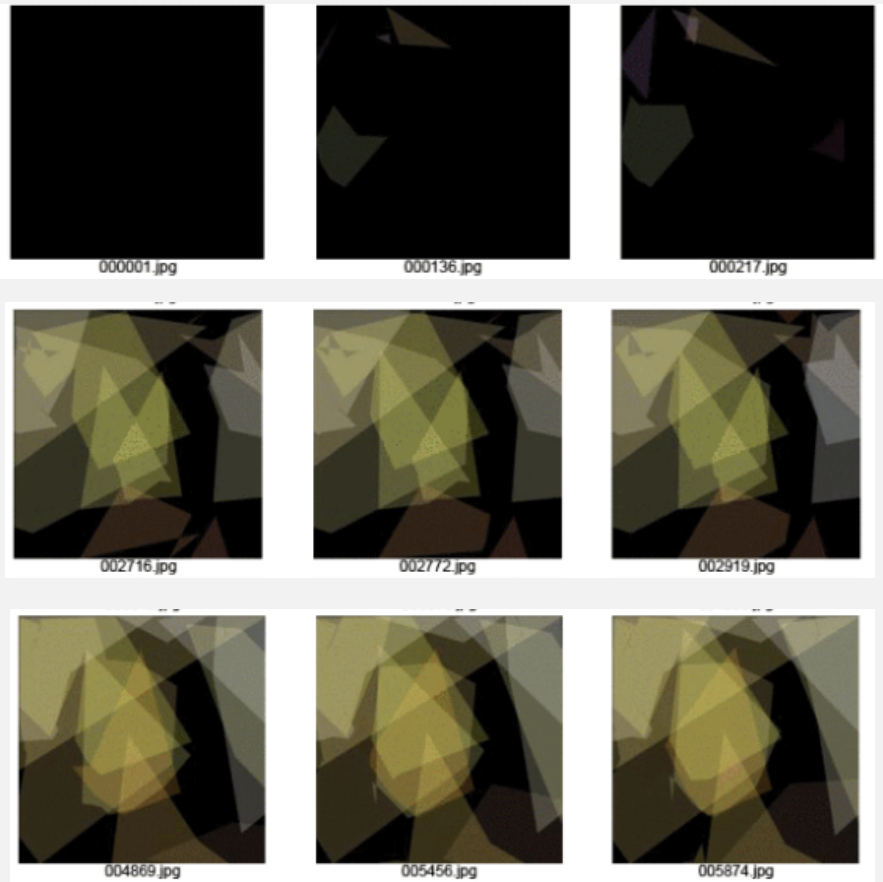


http://en.wikipedia.org/wiki/Travelling_salesman_problem

Approximation

Approach 1: Approximate Heuristics

- Accept incorrect answers
  - Polygon approximation problem: Find colored 50 triangles that best represent an image. Use genetic algorithm to make choice.

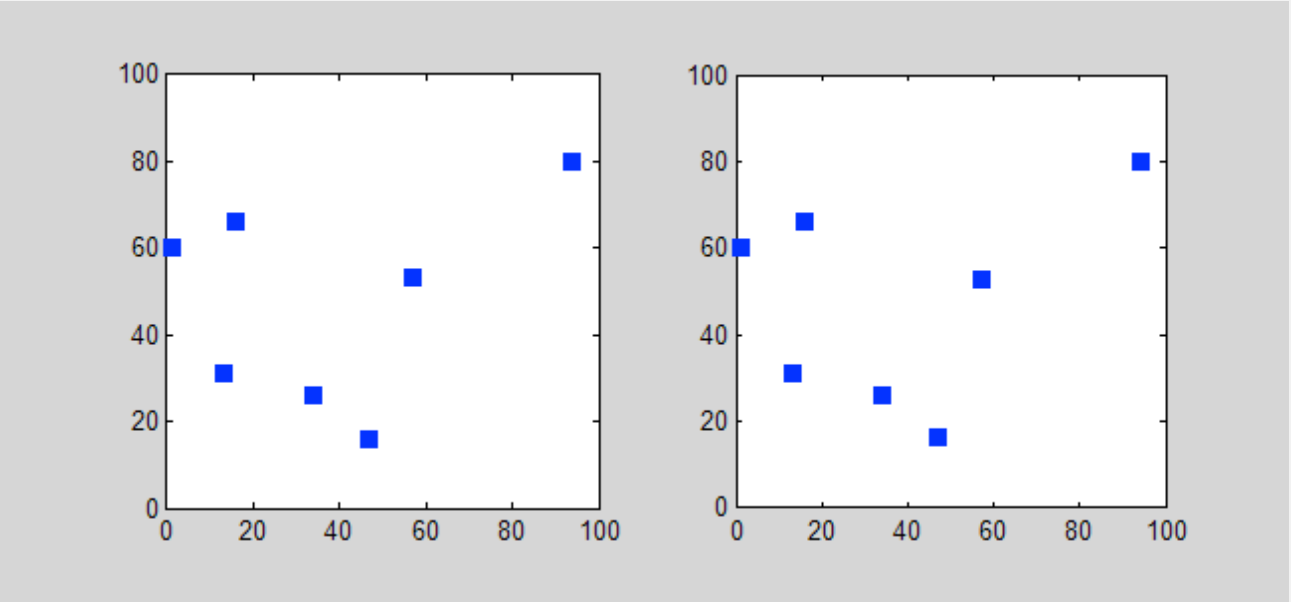


http://rogeralsing.com/2008/12/07/genetic-programming-evolution-of-mona-lisa/

Smarter Searching

Approach 2: Exact Heuristics

- Use a smarter (but still worst case intractable) solution technique
  - TSP: Brute force, but rule out clearly inferior solutions to best known solutions (branch and bound)



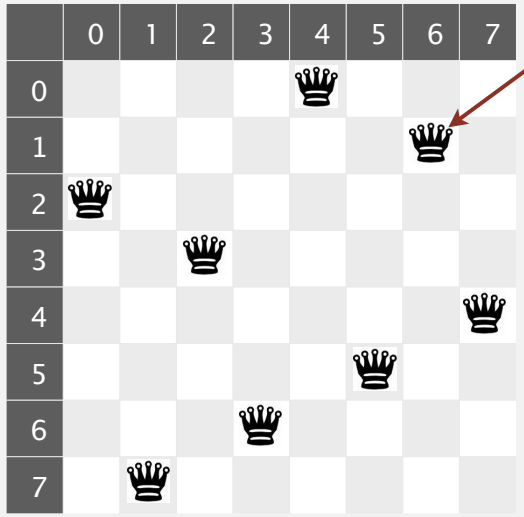
Checks 129 out of 360 possible permutations

http://en.wikipedia.org/wiki/Travelling_salesman_problem

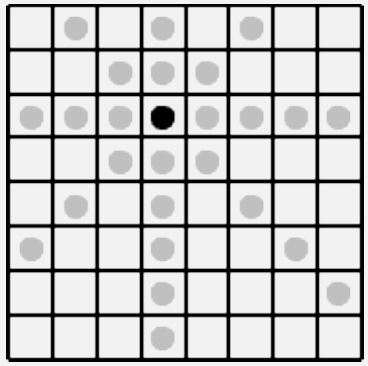
Decision problem is not in NP

Smarter Searching: N-queens problem

Q. How many ways are there to place N queens on an N-by-N board so that no queen can attack any other?



a[1] = 6 means the queen from row 1 is in column 6



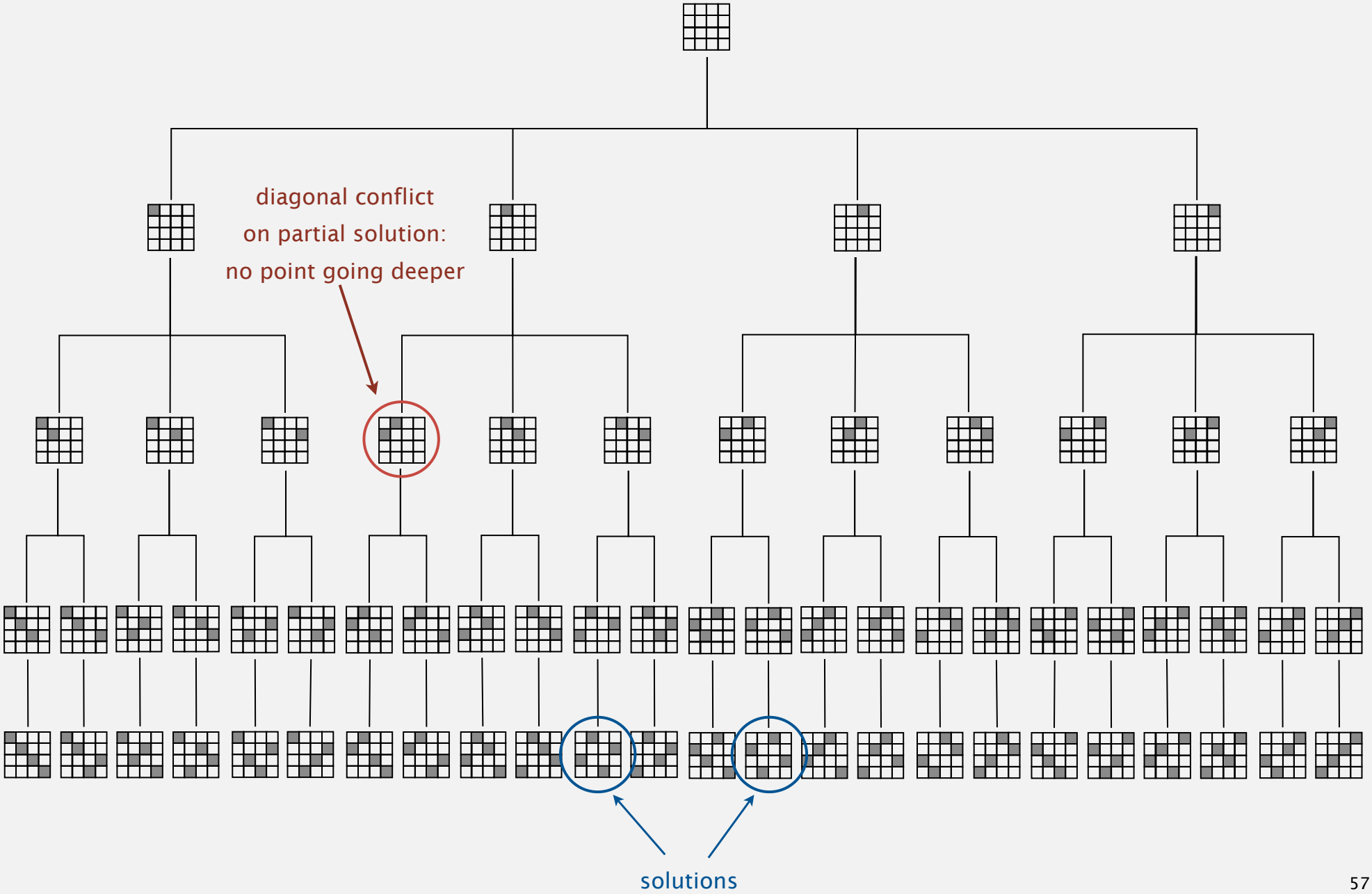
```
int[] a = { 2, 7, 3, 6, 0, 5, 1, 4 };
```

Representation. No 2 queens in the same row or column ⇒ permutation.

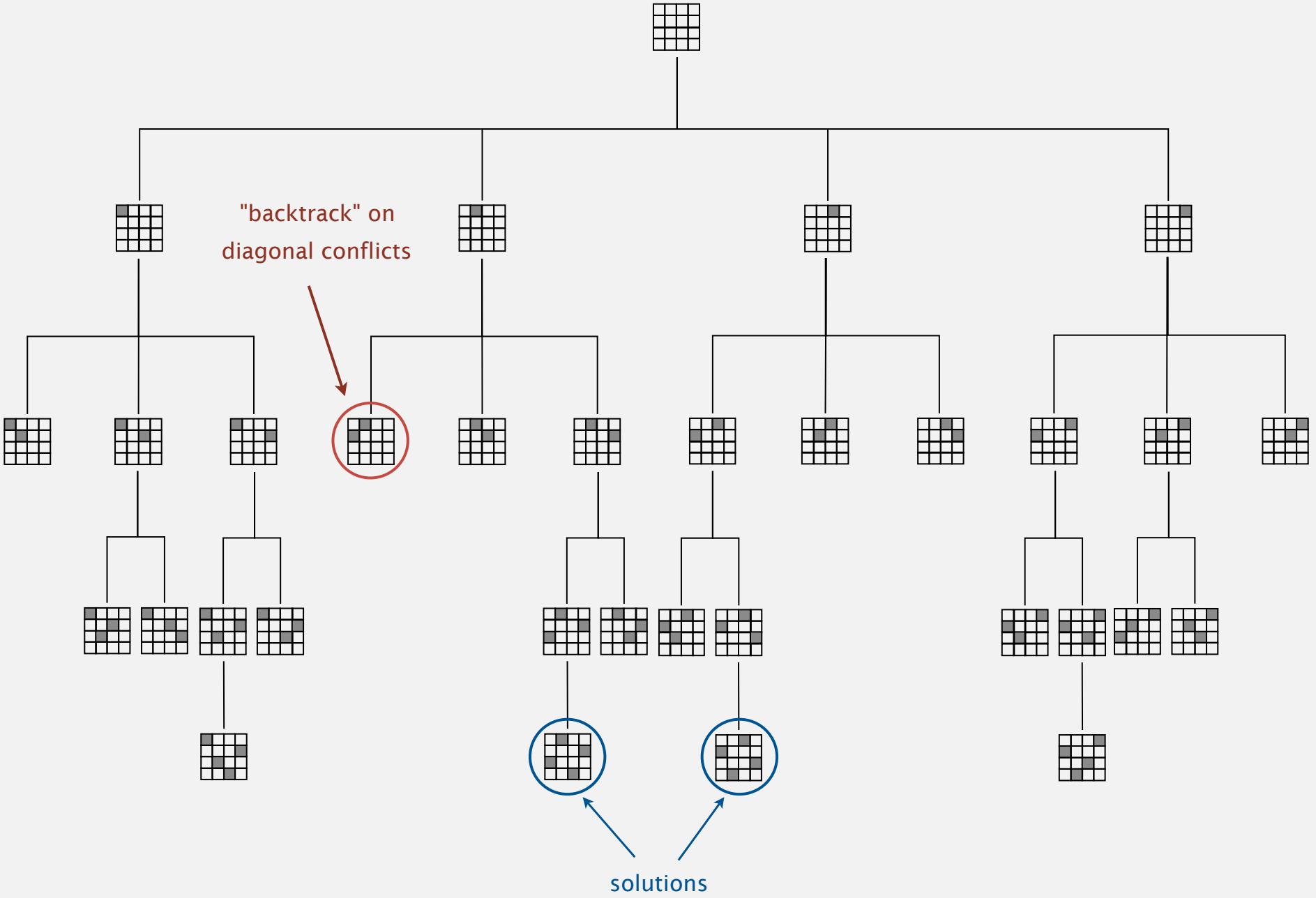
Additional constraint. No diagonal attack is possible.

Challenge. Enumerate (or even count) the solutions. ← unlike N-rooks problem, nobody knows answer for N > 30

4-queens search tree



4-queens search tree (pruned)



Wake up

Approach 3:

- Wake up and realize that it was all just a dream.

Dealing with intractable problems

Approach 4: Take advantage of special structure

- Realize that your problem is actually a special, solvable case.
  - Example 1: Actually in P.
  - Example 2: Worse than P, but only a little.

Examples

- Weighted independent set problem
  - NP-Complete in general.
  - Belongs to P if the graph is a straight line or tree.

http://algs4.cs.princeton.edu

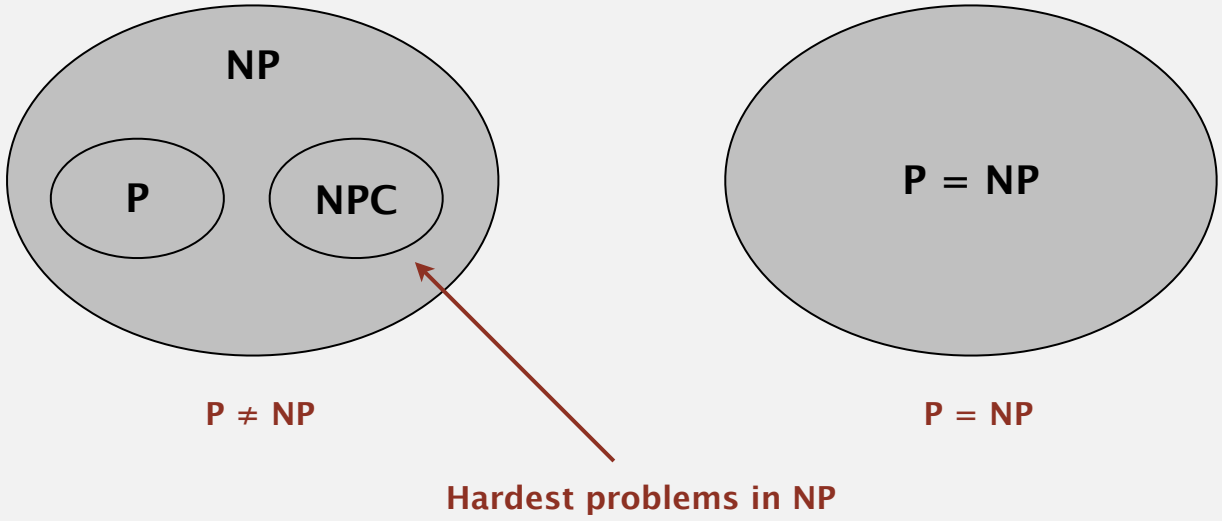
Beyond 226

- *Intro*
- *Reductions*
- *NP completeness*
- *Dealing with NP (or harder) problems*
- *P=NP*
- *Beyond complexity classes*

P = NP?

Does P = NP?

- Equivalently: Is any NP Complete problem also in P?
- Equivalently: Efficiently verifiable ⇒ efficiently solvable?

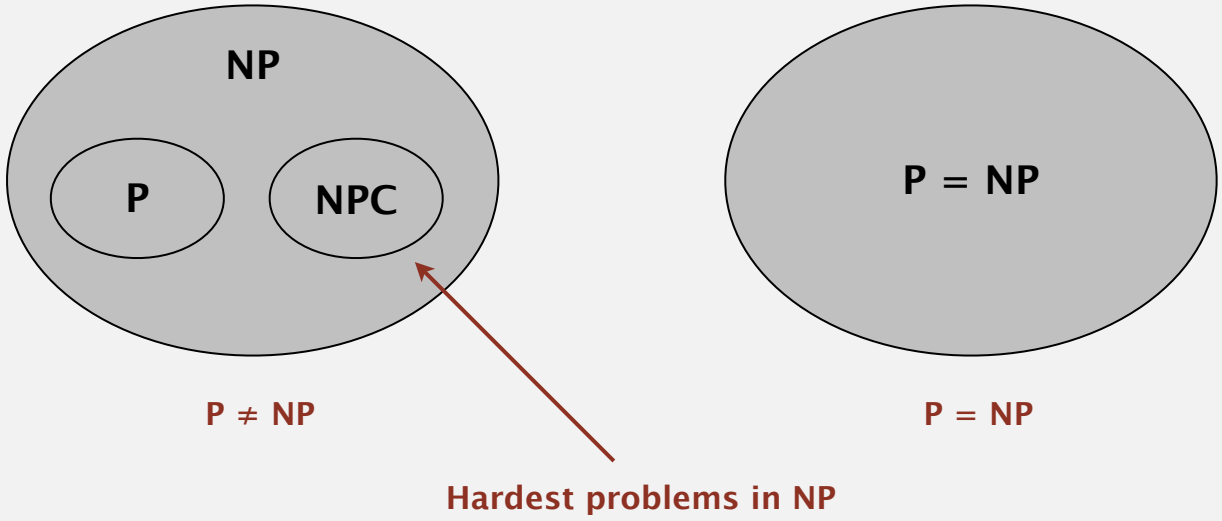


Reminder: NP may as well have been called VP for "Verifiable in Polynomial Time"

P = NP?

Does P = NP?

- Equivalently: Is any NP Complete problem also in P?
- Equivalently: Efficiently verifiable ⇒ efficiently solvable?



Why is P considered efficient? Why is exponential time inefficient?

- n^10000?
- 2^(1.00000000000000001)?
- Known solutions to practical problems simply don't look like this.

P = NP?

Consensus Opinion (Bill Gasarch poll, 2002)

- 9 - Yes
- 61 - No
- 4 - Independent of axiomatic systems typically used in considering the problem.

Why is opinion generally negative?

- Someone would have proved it by now.
  - "The only supporting arguments I can offer are the failure of all efforts to place specific NP-complete problems in P by constructing polynomial-time algorithms." - Dick Karp
  - "For God's sake, let's keep it open for another 100 years! NSF needs to be convinced that theoretical CS is still relevant and supports it." - Ming Li
- Creation of solutions seems philosophically more difficult than verification.
- Mathematical reasons: Way beyond scope of course (and my understanding)

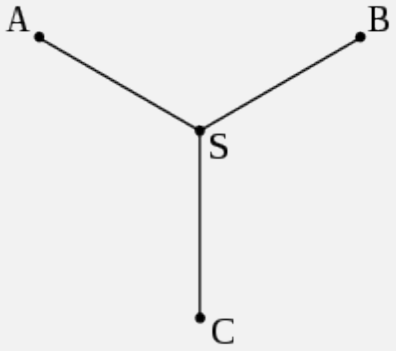
Solving NP Complete Problems

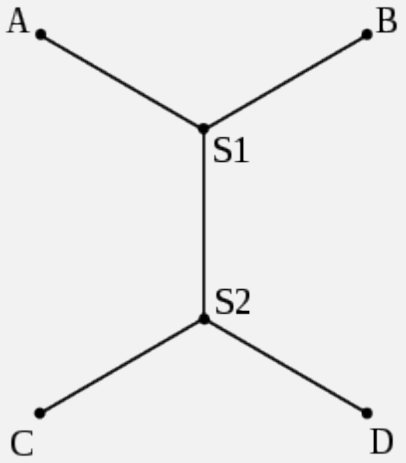
One approach

- Does nature solve any NP complete problems?
- If so, do any such problems admit a polynomial time simulator?

Classic Example

- Soap bubbles and Steiner Trees.





Like Minimum Spanning Trees but you can make up new nodes!

Soap Bubbles and Steiner Trees

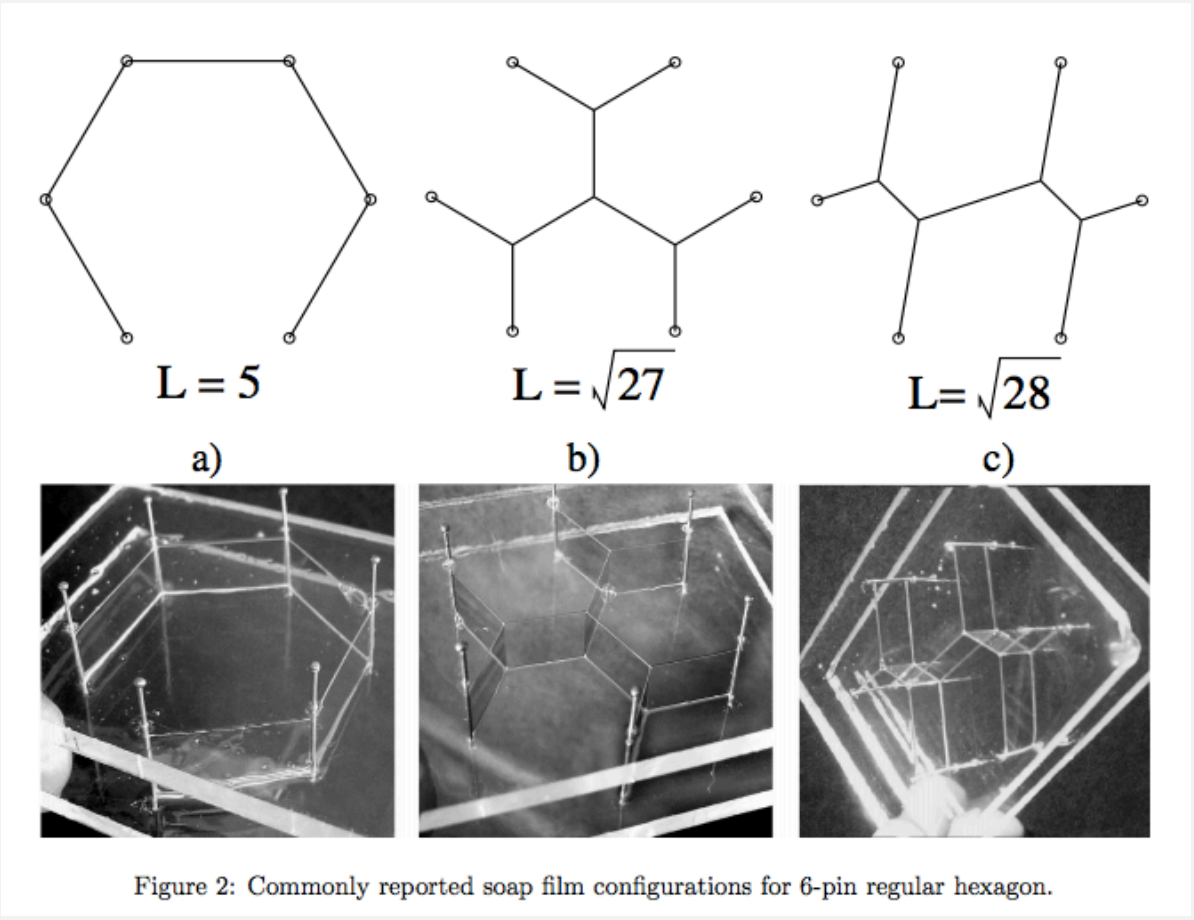


Figure 2: Commonly reported soap film configurations for 6-pin regular hexagon.

http://arxiv.org/pdf/0806.1340v1.pdf

Soap Bubbles mean P=NP?

> *The Clay Mathematics Institute offers a $1 million prize for a solution to the P=?NP problem. We look forward to receiving our award — but concede that the expected format of a solution is an object-level proof, not a meta-level argument like what we provide...*
>
> *... the fact of the matter is that the analog process we exploit is a painfully simple macroscopic phenomenon — as we say, a "normal" physical process. The burden of proof is surely on those who would maintain that the formal machinery of digital physics is insufficient to model something as straightforward as submerging nails in, and retrieving them from, a bucket of soapy water.." — Bringsjord and Taylor (2004)*

http://arxiv.org/pdf/cs/0406056.pdf

The problem

- Soap bubbles aren't always right.
  - Fall into local optima just like heuristic approaches to finding Steiner trees.

Protein Folding

The HP model of protein folding is NP-complete.

- Incorrect but useful analogy: Imagine a shoelace with a bunch of positive and negative charges stuck to various points.
- Simulation tries to figure out final state of charged shoelace if you shake it and let it float into space.

*

Mistake in simulation

- Wrong shape.
- Wrong function prediction.

Mistakes occur in nature

- Bad news for cows (mad cow disease!)
- Bad news for computing (Billions of years of evolution. No solution!)
- Bad news for you.

*: http://themisadventuresofamisplacedalaskan.blogspot.com/2012/04/mad-cow-strikes-back.html

But what if P = NP?

"[A linear or quadratic-time procedure for what we now call NP-complete problems would have] consequences of the greatest magnitude. [For such an procedure] would clearly indicate that, despite the unsolvability of the Entscheidungsproblem, the mental effort of the mathematician in the case of yes-or-no questions could be completely replaced by machines." — *Kurt Gödel*

One of these things, is not like the other..

Millenium Prize Problems

- Hodge Conjecture
- Poincare Conjecture (solved!)
- Riemann Hypothesis
- Yang-Mills existence and mass gap
- Navier-Stokes existence and smoothness
- Birch and Swinnerton-dyer conjecture
- P=NP
  - If true, proof might allow you to trivially solve all of the other problems.

What if P = NP?

"I have heard it said, with a straight face, that a proof of P = NP would be important because it would let airlines schedule their flights better, or shipping companies pack more boxes in their trucks!

If [P = NP], then we could quickly find the smallest Boolean circuits that output (say) a table of historical stock market data, or the human genome, or the complete works of Shakespeare. It seems entirely conceivable that, by analyzing these circuits, we could make an easy fortune on Wall Street, or retrace evolution, or even generate Shakespeare's 38th play. For broadly speaking, that which we can compress we can understand, and that which we can understand we can predict.

So if we could solve the general case—if knowing something was tantamount to knowing the shortest efficient description of it—then we would be almost like gods. [Assuming P ≠ NP] is the belief that such power will be forever beyond our reach." — Scott Aaronson

http://www.scottaaronson.com/papers/npcomplete.pdf

Great paper: Also where I stole the ideas about soap bubbles and protein folding.

http://algs4.cs.princeton.edu

6.5 Reductions

- *Intro*
- *Reductions*
- *NP completeness*
- *Dealing with NP (or harder) problems*
- *P=NP*
- *Beyond complexity classes*

Can machines think?

AI-Complete problems

- Communication in natural language
- Translation of natural language
- Writing a play
- Programming
- Theorem proving (can be formalized into an NP complete problem)

Can machines think?

AI-Complete problems

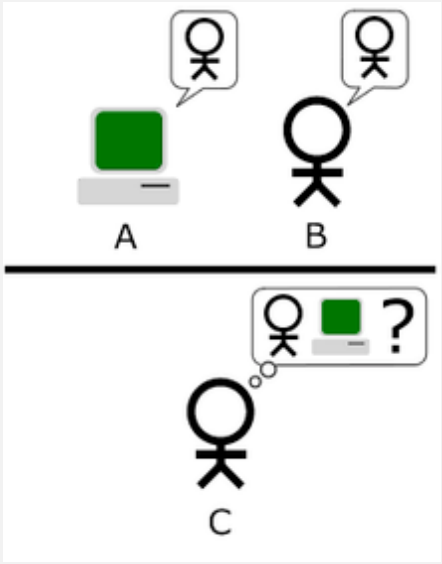
- Communication in natural language
- Translation of natural language
- Writing a play
- Programming
- Theorem proving (can be formalized into an NP complete problem)

Proof of concept

- Brains
- Bad API

The Turing test

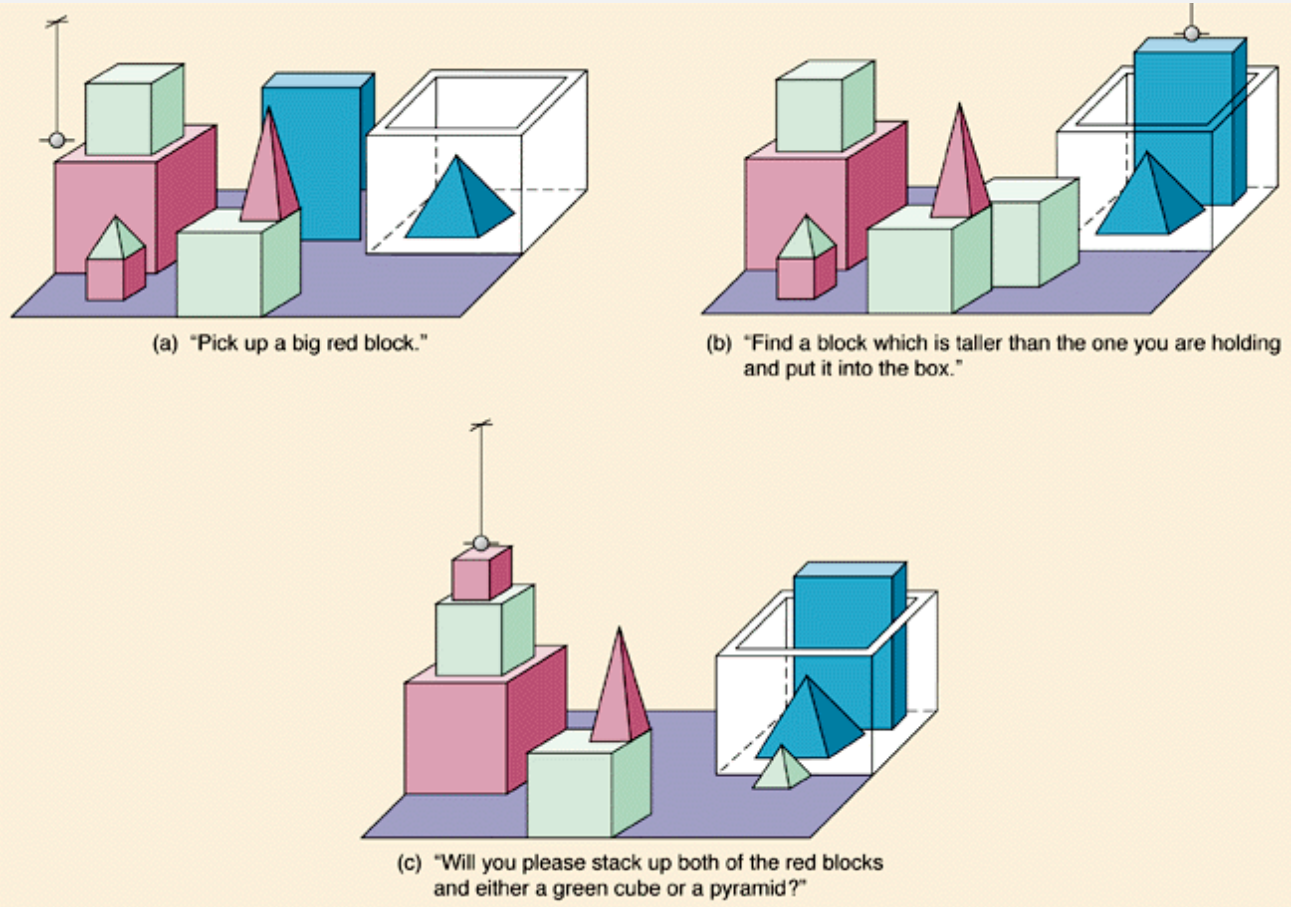
Separate the scientific question from the philosophical question



http://cmst1a0970733.blogspot.com/2009/10/collective-computer-intelligence-mark.html

http://www.cosc.canterbury.ac.nz/csfieldguide/student/Artificial%20intelligence.html

A brief history of AI



http://www.wiley.com/college/busin/icmis/oakman/outline/chap11/slides/blocks.htm

A brief history of AI

The spirit is willing but the flesh is weak

(to Russian and back)

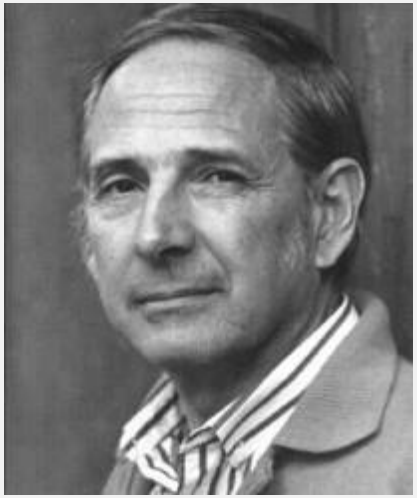
A brief history of AI

The ascendancy of data and machine learning

Searle's Chinese Room

John Searle

- Incredibly famous professor in the philosophy of mind (and language).
- Goal:
  - Debunk the Turing test (and similar) as proof of intelligence.



Searle's Chinese Room

Imagine a box with a computer inside.

- You slip messages in Chinese into a slot in a box.
- The machine reads the Chinese, processes it according to some set of rules, and prints out a response.
- The Chinese room behaves exactly as if it were a native Chinese speaker.
- Does the machine truly understand Chinese (Strong AI), or is it just simulating understanding (weak AI)?

我爸是李刚!

Karl Gottlieb von Windisch's 1784 book *Inanimate Reason*

Searle's Chinese Room

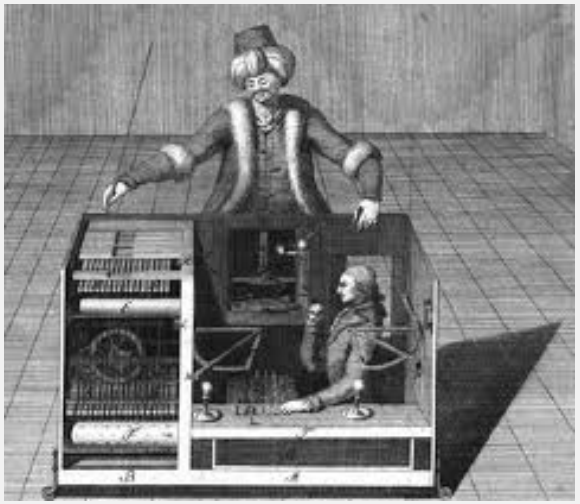
Observation

- The human mind can simulate a Turing machine (equivalent to: it is possible to manually simulate Java code).

Searle's Chinese Room

Imagine a box with a guy inside.

- You slip messages in Chinese to a monolingual Anglophone in a box.
- The guy has:
  - A printout of the source code of the Chinese room computer program.
  - An enormous supply of pencils, erasers, paper, and filing cabinets.
- When he receives a message, the guy mechanically follows the rules, writes out the resulting response, and slips it back to you.
- Since the guy in the box doesn't understand Chinese (or what he's saying), then the computer doesn't either: it's just simulating intelligence.

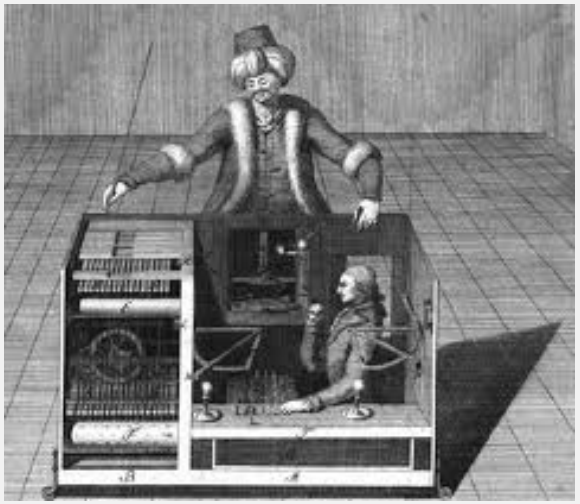


Karl Gottlieb von Windisch's 1784 book *Inanimate Reason*

Josh's Searle's Chinese Room

Imagine a box with a humanoid robot inside.

- You slip messages in Chinese to a humanoid robot inside a box.
- The robot has:
  - A printout of the source code of the Chinese room computer program.
  - An enormous supply of pencils, erasers, paper, and filing cabinets.
  - Minimal complexity: Ability to OCR characters and follow rules in book.
- When he receives a message, the robot mechanically follows the rules, writes out the resulting response, and slips it back to you.
- Does the humanoid robot understand Chinese?
  - My opinion: No, and it's irrelevant!



The Chinese Room

What the Chinese Room Purports to Prove

- The Turing Test doesn't prove intelligence.
- Argument:
  - Turing: If a digital computer mechanically generates good conversation, then the computer IS sentient.
  - Searle: Since a man can unthinkingly generate good conversation according to same mechanical rules, then a computer doing the same thing is NOT sentient (because the man wasn't thinking).

- My opinion: The man in the Chinese Room is acting as a non-sentient component of a sentient system.
  - The 'smarts' of the system are all in the rulebook (which is going to be fantastically complex).

> *"[The field of cognitive science ought to be redefined as] the ongoing research program of showing Searle's Chinese Room Argument to be false."* — *Pat Hayes*

Can humans think?

Consequences if the Chinese Room (or similar) arguments are right

- Just because YOUR brain is providing me with a good conversation doesn't prove anything!
- You might just be pretending to be sentient.
  - Philosophical zombie.

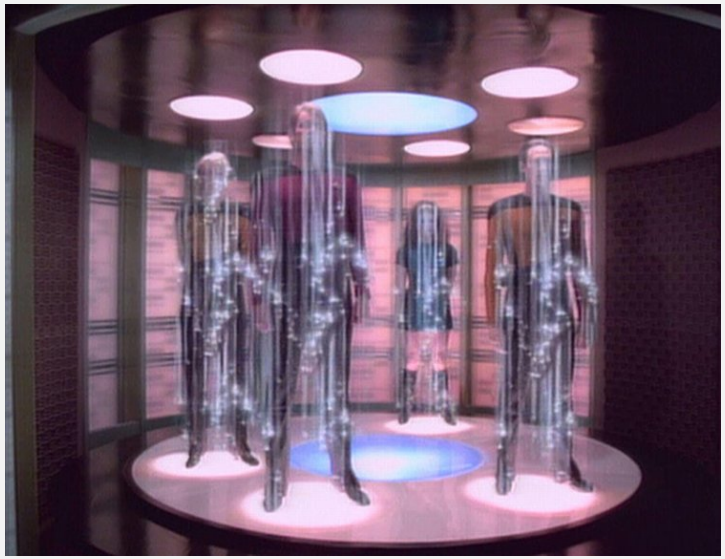
> *"**Intellect:** By convention there is sweetness, by convention bitterness, by convention color, in reality only atoms and the void.*
>
> ***Senses:** Foolish intellect! Do you seek to overthrow us, while it is from us that you take your evidence?" — Democritus (circa 450 BC), dialogue between the intellect of senses*

Quote by way of Scott Aaronson (in turn by way of Schrodinger)

The Hug Paradox

Josh Hug believes brains are just mechanical machines.

- Josh Hug would also not volunteer to be deconstructed and then reconstructed.
  - He would be perfectly willing to have his brains cells replaced by precise electronic analogues, one by one, at a reasonable rate (say entire brain in one day).
  - If replacement process was too fast (picoseconds?): Hell no!
  - There is clearly an error in Josh Hug's thinking.



http://us.123rf.com/400wm/400/400/rbhavana/rbhavana1012/rbhavana101200549/8368432-neuron-in-isolated-background.jpg

Deep into the fringes

The Chinese Lookup Table

- Suppose we have an enormous lookup table that encodes every possible stream of Chinese conversation of duration 20 years or less.
- Features:
  - More atoms than the universe (obscenely so!).
  - Indistinguishable from a sentient being (for 20 years at least).
- Would such a lookup table think?

Things to ponder

- Kolmogorov complexity is massive compared to a representation of human intelligence.
  - Consciousness as compression [??].
- Indices to lookup table are integer representations of the entire conversation!
- Seemingly simulation of intelligent entity to generate deterministically.
  - Static consciousness [??].
- Chance of generating a good lookup table randomly is non-zero!

Can the Universe solve NP-complete problems?

More generally, what's the computational complexity class of the Universe?

- Soap bubbles, DNA, etc?
  - Probably not
- Quantum?
  - Maybe
  - Roger Penrose thinks brains are non-algorithmic because of quantum effects
- Time travel? Black holes?
  - Whoa! Duuuuuuddddddeee....

Is the Universe a simulation?