

# NP-Completeness



# Properties of Algorithms

A given problem can be solved by many different algorithms. Which **ALGORITHMS** will be useful in practice?

A working definition: (Jack Edmonds, 1962)

- Efficient: polynomial time for ALL inputs.
- Inefficient: "exponential time" for SOME inputs.

Robust definition has led to explosion of useful algorithms for wide spectrum of problems.

- Notable exception: simplex algorithm.

# Exponential Growth

Exponential growth dwarfs technological change.

- Suppose each electron in the universe had power of today's supercomputers.
- And each works for the life of the universe in an effort to solve TSP problem using  $N!$  algorithm.

Some Numbers	
Quantity	Number
Home PC instructions / second	$10^9$
Supercomputer instructions / second	$10^{12}$
Seconds per year	$10^9$
Age of universe †	$10^{13}$
Electrons in universe †	$10^{79}$

† Estimated

- Will not succeed for 1,000 city TSP!

$$1000! \gg 10^{1000} \gg 10^{79} \times 10^{13} \times 10^9 \times 10^{12}$$

# Properties of Problems

Which **PROBLEMS** will we be able to solve in practice?

- Those with efficient algorithms.
- How can I tell if I am trying to solve such a problem?

 **Theory of NP-completeness helps.**

Yes	Probably No
Shortest path	Longest path
Euler cycle	Hamiltonian cycle
Min cut	Max cut
2-SAT	3-SAT
PLANAR-2-COLOR	PLANAR-3-COLOR
PLANAR-4-COLOR	PLANAR-3-COLOR
Matching	3D-Matching
Baseball elimination	Soccer elimination
Bipartite vertex cover	Vertex cover

Unknown
Primality
Factoring
Graph isomorphism

# P

## Decision problem X.

- X is a (possibly infinite) set of binary strings.
- Instance: finite binary string  $s$ , of length  $|s|$ .
- Algorithm A solves X if  $A(s) = \text{YES} \Leftrightarrow s \in X$ .

## Polynomial time.

- Algorithm A runs in polynomial-time if for every instance  $s$ , A terminates in at most  $p(s)$  "steps", where  $p$  is some polynomial.

## Definition of P.

- Set of all **decision problems** solvable in **polynomial time** on a deterministic Turing machine.

## Examples:

- **MULTIPLE**: Is the integer  $y$  a multiple of  $x$ ?
- **RELPRIME**: Are the integers  $x$  and  $y$  relatively prime?
- **PERFECT-MATCHING**: Given graph  $G$ , is there a perfect matching?

# Strong Church-Turing Thesis

Definition of P fundamental because of SCT.

**Strong Church-Turing thesis:**

- P is the set of decision problems solvable in polynomial time on **REAL** computers.

**Evidence supporting thesis:**

- True for all physical computers.
- Can create deterministic TM that efficiently simulates any real general-purpose machine (and vice versa).

**Possible exception?**

- Quantum computers: no conventional gates.

# Efficient Certification

## Certification algorithm.

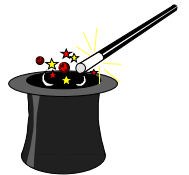
- Design an algorithm that checks whether proposed solution is a YES instance.

## Algorithm C is an efficient certifier for X if:

- C is a polynomial-time algorithm that takes two inputs s and t.
- There exists a polynomial  $p(\ )$  so that for every string s,  $s \in X \Leftrightarrow$  there exists a string t such that  $|t| \leq p(|s|)$  and  $C(s, t) = \text{YES}$ .

## Intuition.

- Efficient certifier views things from "managerial" viewpoint.
- It doesn't determine whether  $s \in X$  on its own.
- Rather, it evaluates a proposed proof t that  $s \in X$ .
- Accepts if and only if given a "short" proof of this fact.

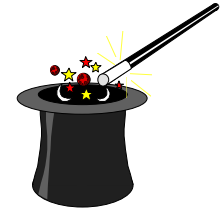


# Certifiers and Certificates

**COMPOSITE:** Given integer  $s$ , is  $s$  composite?

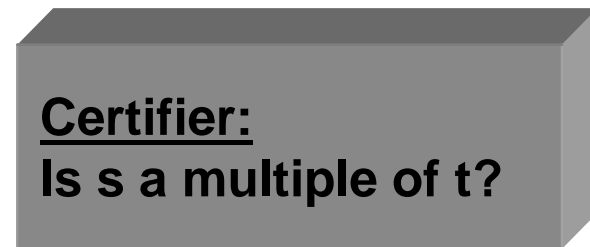
**Observation.**  $s$  is composite  $\Leftrightarrow$  there exists an integer  $1 < t < s$  such that  $s$  is a multiple of  $t$ .

- YES instance:  $s = 437,669$ .
  - certificate  $t = 541$  or  $809$  (a factor)



Input  $s$ :  
437,669

Certificate  $t$ :  
541



YES

NO

$s$  is a YES instance

no conclusion

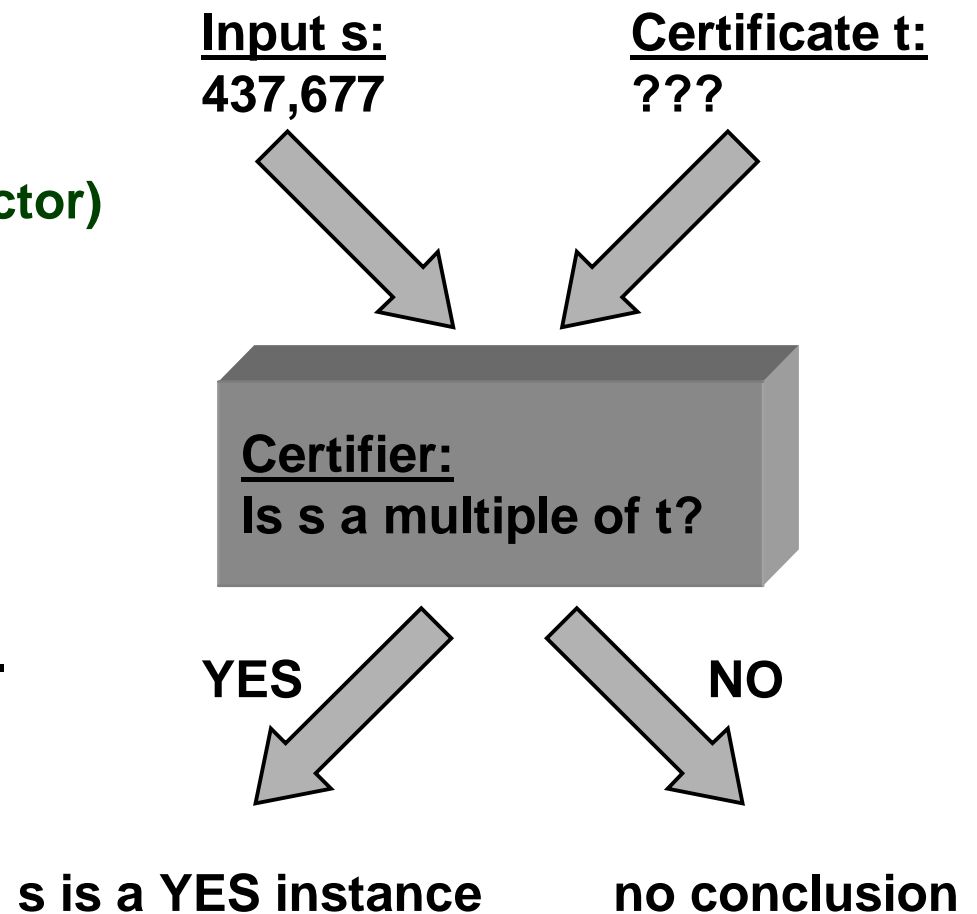


# Certifiers and Certificates

**COMPOSITE:** Given integer  $s$ , is  $s$  composite?

**Observation.**  $s$  is composite  $\Leftrightarrow$  there exists an integer  $1 < t < s$  such that  $s$  is a multiple of  $t$ .

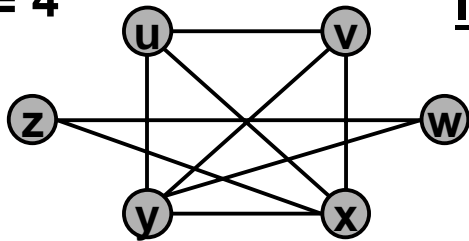
- **YES instance:**  $s = 437,669$ .
  - certificate  $t = 541$  or  $809$  (a factor)
- **NO instance:**  $s = 437,677$ .
  - no witness can fool verifier into saying YES
- **Conclusion:** **COMPOSITE**  $\in$  NP.



# Certifiers and Certificates

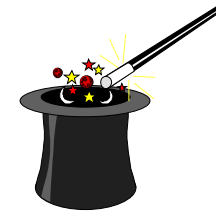
**CLIQUE:** Given an undirected graph, is there a subset  $S$  of  $k$  nodes such that there is an arc connecting every pair of nodes in  $S$ ?

$k = 4$



Input s:

Certificate t:  
 $\{u, v, w, x\}$



Certifier:

1. For all pairs of nodes  $v$  and  $w$  in  $t$ , check that  $(v, w)$  is an edge in the graph.
2. Check that number of nodes in  $t \leq k$ .

YES

NO

**CLIQUE  $\in$  NP.**

**s is a YES instance**

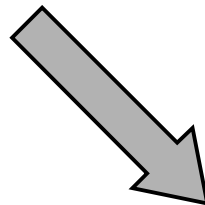
**no conclusion**

# Certifiers and Certificates

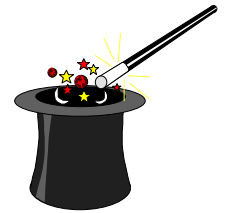
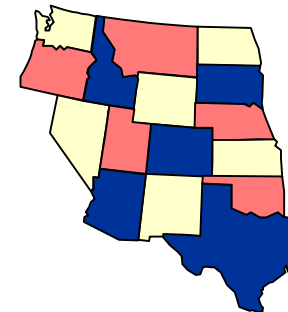
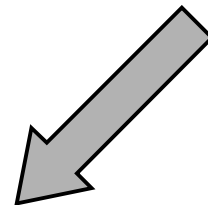
**3-COLOR:** Given planar map, can it be colored with 3 colors?



Input s:



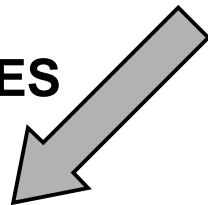
Certificate t:



Certifier:

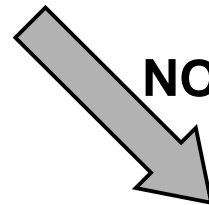
1. Check that s and t describe same map.
2. Count number of distinct colors in t.
3. Check all pairs of adjacent states.

YES



s is a YES instance

NO



no conclusion

**3-COLOR  $\in$  NP.**

# NP

## Definition of NP:

- Does NOT mean "not polynomial."

## Definition of NP:

- Set of all decision problems for which there exists an efficient certifier.
- Definition important because it links many fundamental problems.

**Claim:**  $P \subseteq NP$ .

**Proof:** Consider problem  $X \in P$ .

- Then, there exists efficient algorithm  $A(s)$  that solves  $X$ .
- Efficient certifier  $B(s, t)$ : return  $A(s)$ .

# NP

## Definition of EXP:

- Set of all decision problems solvable in **exponential** time on a deterministic Turing machine.

**Claim:**  $NP \subseteq EXP$ .

**Proof:** Consider problem  $X \in NP$ .

- Then, there exists efficient certifier  $C(s, t)$  for  $X$ .
- To solve input  $s$ , run  $C(s, t)$  on all strings  $t$  with  $|t| \leq p(|s|)$ .
- Return YES, if  $C(s, t)$  returns YES for any of these.

## Useful alternate definition of NP:

- Set of all decision problems solvable in polynomial time on a **NONDETERMINISTIC** Turing machine.
- Intuition: act of searching for  $t$  is viewed as a non-deterministic search over the space of possible proofs. Nondeterministic TM can try all possible solutions in parallel.

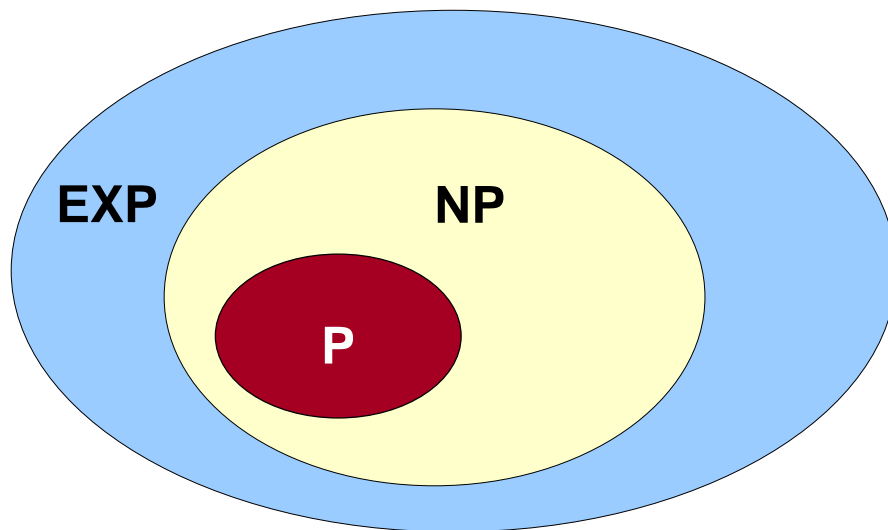
# The Main Question

Does  $P = NP$ ? (Edmonds, 1962)

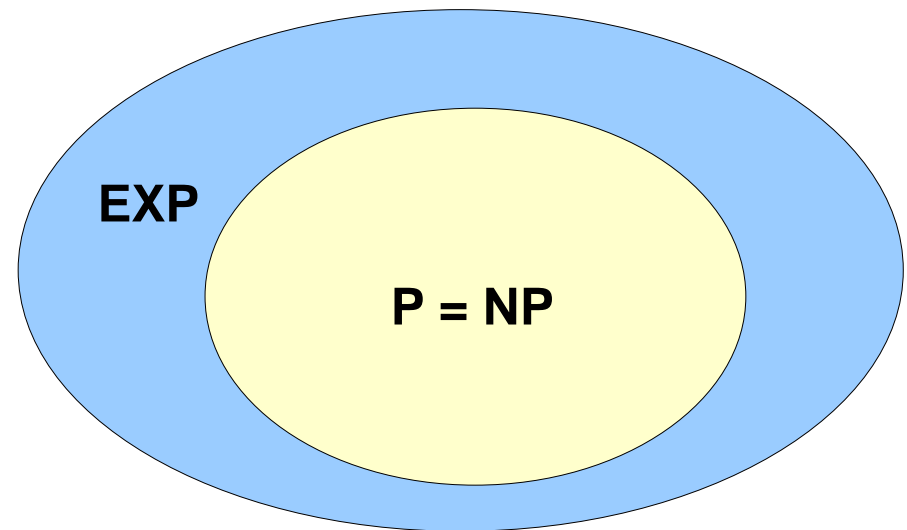
- Is the original **DECISION** problem as easy as **CERTIFICATION**?
- Does nondeterminism help you solve problems faster?

Most important open problem in computer science.

- If yes, staggering practical significance.
- Clay Foundation Millennium \$1 million prize.



If  $P \neq NP$



If  $P = NP$

# The Main Question

## Generator (P)

- Factor integer  $s$ .
- Color a map with minimum # colors.
- Design airfoil of minimum drag.
- Prove a beautiful theorem.
- Write a beautiful sonnet.
- Devise a good joke.
- Vinify fine wine.
- Orate a good lecture.
- Ace an exam.

## Certifier (NP)

- Is  $s$  a multiple of  $t$ ?
- Check if all adjacent regions have different colors.
- Compute drag of airfoil.
- Understand its proof.
- Appreciate it.
- Laugh at it.
- Be a wine snob.
- Know when you've heard one.
- Verify TA's solutions.

**Imagine the wealth of a society that produces optimal planes, bridges, rockets, theorems, art, music, wine, jokes.**

# The Main Question

Does  $P = NP$ ?

- Is the original **DECISION** problem as easy as **CERTIFICATION**?

If yes, then:

- Efficient algorithms for 3-COLOR, TSP, FACTOR, . . .
- Cryptography is impossible (except for one-time pads) on conventional machines.
- Modern banking system will collapse.
- Harmonial bliss.

If no, then:

- Can't hope to write efficient algorithm for TSP.
- But maybe efficient algorithm still exists for FACTOR . . . .



# The Main Question

Does  $P = NP$ ?

- Is the original **DECISION** problem as easy as **CERTIFICATION**?

Probably no, since:

- Thousands of researchers have spent four frustrating decades in search of polynomial algorithms for many fundamental NP problems without success.
- Consensus opinion:  $P \neq NP$ .

But maybe yes, since:

- No success in proving  $P \neq NP$  either.

# Polynomial Transformation

Problem X **polynomial reduces (Cook-Turing)** to problem Y ( $X \leq_p Y$ ) if arbitrary instances of problem X can be solved using:

- Polynomial number of standard computational steps, plus
- Polynomial number of calls to oracle that solves problem Y.

Problem X **polynomial transforms (Karp)** to problem Y if given any input  $x$  to X, we can construct an input  $y$  such that  $x$  is a YES instance of X if and only if  $y$  is a YES instance of Y.

- We require  $|y|$  to be of size polynomial in  $|x|$ .

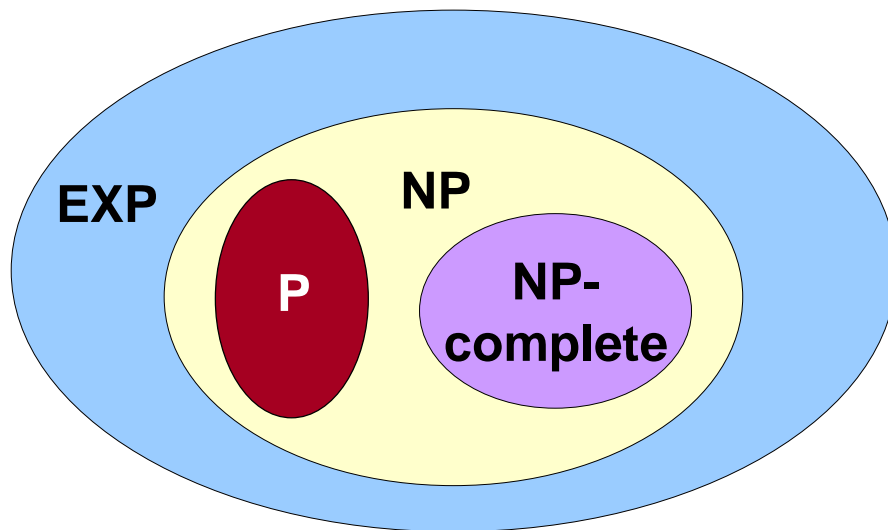
**Polynomial transformation is polynomial reduction with just one call to oracle for Y, exactly at the end of the algorithm for X.**

**Note: all previous reductions were of this form!**

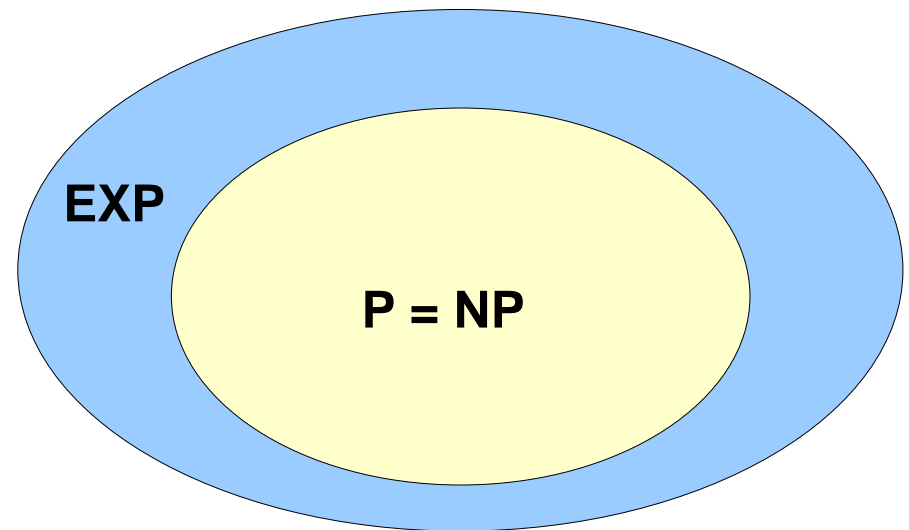
# NP-Complete

## Definition of NP-complete:

- A problem Y in NP with the property that for every problem X in NP, X polynomial transforms to Y.
- "Hardest computational problems" in NP.



If  $P \neq NP$



If  $P = NP$

# NP-Complete

## Definition of NP-complete:

- A problem Y in NP with the property that for every problem X in NP, X polynomial transforms to Y.

## Significance.

- Efficient algorithm for any NP-complete problem  $\Rightarrow$  efficient algorithm for every other problem in NP.
- Links together a huge and diverse number of fundamental problems:
  - TSP, 3-COLOR, CNF-SAT, CLIQUE, . . . . .
- Can implement any computer program in 3-COLOR.

## Notorious complexity class.

- Only exponential algorithms known for these problems.
- Called "**intractable**" - unlikely that they can be solved given limited computing resources.

# Some NP-Complete Problems

Most natural problems in NP are either in P or NP-complete.

Six basic genres and paradigmatic examples of NP-complete problems.

- **Packing problems:** SET-PACKING, INDEPENDENT-SET.
- **Covering problems:** SET-COVER, VERTEX-COVER.
- **Sequencing problems:** HAMILTONIAN-CYCLE, TSP.
- **Partitioning problems:** 3-COLOR, CLIQUE.
- **Constraint satisfaction problems:** SAT, 3-SAT.
- **Numerical problems:** SUBSET-SUM, PARTITION, KNAPSACK.

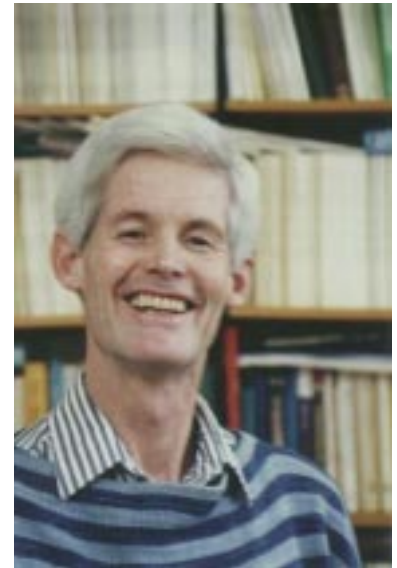
**Caveat:** PRIME, FACTOR not known to be NP-complete.

# The "World's First" NP-Complete Problem

**CNF-SAT is NP-complete. (Cook-Levin, 1960's)**

## Idea of proof:

- **Given problem X in NP, by definition, there exists nondeterministic TM M that solves X in polynomial time.**
- **Possible execution of M on input string s forms a branching tree of configurations, where each configuration gives snapshot of M (tape contents, head position, state of finite control) at some time step t.**
- **M is polynomial time  $\Rightarrow$  polynomial tree depth  $\Rightarrow$  polynomial number of tape cells in play.**
- **Use polynomial number of Boolean variables to model which symbol occupies cell i at time t, location of read head at time t, state of finite control, etc.**
- **Use polynomial number of clauses to ensure machine makes legal moves, starts and ends in appropriate configurations, etc.**



**Stephen Cook**

# Establishing NP-Completeness

## Definition of NP-complete:

- A problem  $Y \in \text{NP}$  with the property that for every problem  $X$  in NP,  $X$  polynomial transforms to  $Y$ .

**Cook's theorem.** CNF-SAT is NP-complete.

## Recipe to establish NP-completeness of problem $Y$ .

- Step 1. Show that  $Y \in \text{NP}$ .
- Step 2. Show that CNF-SAT (or any other NP-complete problem) transforms to  $Y$ .

## Example: CLIQUE is NP-complete.

- ✓ Step 1. CLIQUE  $\in$  NP.
- ✓ Step 2. CNF-SAT polynomial transforms to CLIQUE.

# Minesweeper Consistency Problem

## Minesweeper.

- **Start: blank grid of squares.**
- **Some squares conceal mines; the rest are safe.**
- **Find location of mines without detonating any.**
- **Choose a square.**
  - **if mine underneath, it detonates and you lose**
  - **If no mine, computer tells you how many total mines in 8 neighboring squares**



# Minesweeper Consistency Problem

Minesweeper consistency problem.

- Given a state of what purports to be a Minesweeper game, is it logically consistent.

	1	2	1	
	1		1	
	1		1	
	2	2	2	2

?	1	?	1	?
?	1	2	1	?
?	1		1	?
	1		1	
	2	2	2	2

?	1	?	1	?
?	1	2	1	?
?	1		1	?
1	1		1	1
	2	2	2	2

?	1	?	1	?
?	1	2	1	?
?	1		1	?
1	1		1	1
	2	2	2	2
?	1	?	1	?

YES

	1	2	1	
	1		1	
	1		1	
	1	2	2	2

?	1	?	1	?
?	1	2	1	?
?	1		1	?
	1		1	
	1	2	2	2

?	1	?	1	?
?	1	2	1	?
?	1		1	?
1	1		1	1
	1	2	2	2

NO

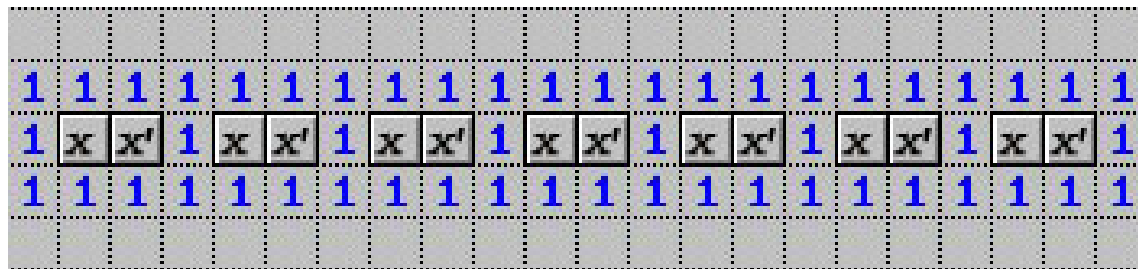
# Minesweeper Consistency Problem

## Minesweeper consistency problem.

- Given a state of what purports to be a Minesweeper game, is it logically consistent.

## Claim. Minesweeper consistency is NP-complete.

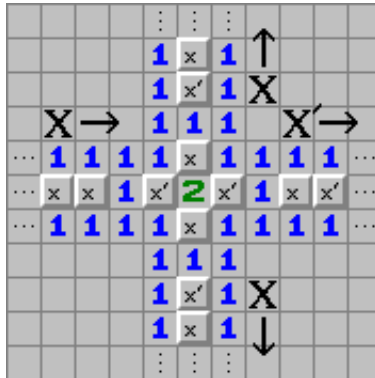
- Proof idea: reduce from circuit satisfiability.
- Build circuit by laying out appropriate minesweeper configurations.



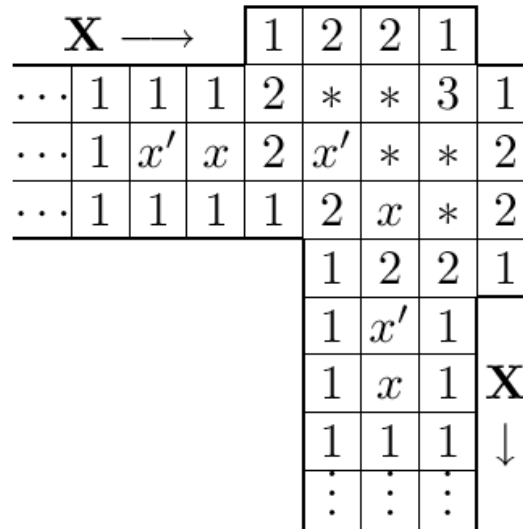
A Minesweeper Wire



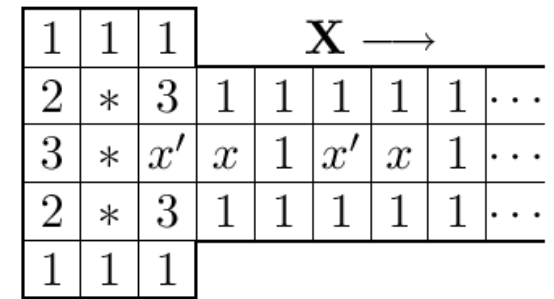
# Minesweeper Consistency Problem



**A 3-way Splitter**



(a)



(b)

Figure 4: (a) A bent wire. (b) A terminated wire.



**A Phase Changer**



# Coping With NP-Completeness

Hope that worst case doesn't occur.

- Complexity theory deals with worst case behavior. The instance(s) you want to solve may be "easy."
  - TSP where all points are on a line or circle
  - 13,509 US city TSP problem solved



(Cook et. al., 1998)

# Coping With NP-Completeness

Hope that worst case doesn't occur.

Change the problem.

- Develop a heuristic, and hope it produces a good solution.
- Design an **approximation algorithm**: algorithm that is guaranteed to find a high-quality solution in polynomial time.
  - active area of research, but not always possible unless  $P = NP$ !
  - Euclidean TSP tour within 1% of optimal
  - stay tuned



Sanjeev Arora (1997)

# Coping With NP-Completeness

**Hope that worst case doesn't occur.**

**Change the problem.**

**Exploit intractability.**

- **Cryptography.**



# Coping With NP-Completeness

**Hope that worst case doesn't occur.**


**Change the problem.**

**Exploit intractability.**




**Keep trying to prove  $P = NP$ .**

# Coping With NP-Completeness

A person can be at most two of the following three things:

-  **Honest.**
-  **Intelligent.**
-  **A politician.**

If a problem is NP-complete, you design an algorithm to do at most two of the following three things:

-  **Solve the problem exactly.**
-  **Guarantee to solve the problem in polynomial time.**
-  **Solve arbitrary instances of the problem.**

# Asymmetry of NP

**Definition of NP:**  $X \in \text{NP}$  if there exists a certifier  $C(s, t)$  such that:

- Input string  $s$  is a YES instance if and only if **there exists** a short certificate  $t$  such that  $C(s, t) = \text{YES}$ .
- Alternatively, input string  $s$  is a NO instance if and only if **for all** short  $t$ ,  $C(s, t) = \text{NO}$ .

**Ex. HAM-CYCLE vs. NO-HAM-CYCLE.**

- Given  $G$  and a proposed Hamiltonian cycle, easy to check if it really is a Hamiltonian cycle.
- Given  $G$ , hard to assert that it is **not** Hamiltonian.

**Ex. PRIME vs. COMPOSITE.**

- Given integer  $s$  and proposed factor  $t$ , it is easy to check if  $s$  is a multiple of  $t$ .
- Appears harder to assert that an integer is **not** composite.

# Co-NP

**NP:** Decision problem  $X \in \text{NP}$  if there exists a certifier  $C(s, t)$  s.t.

- Input string  $s$  is a **YES** instance if and only if **there exists** a short certificate  $t$  such that  $C(s, t) = \text{YES}$ .
- Alternatively, input string  $s$  is a **NO** instance if and only if **for all** short  $t$ ,  $C(s, t) = \text{NO}$ .

**co-NP:**  $X \in \text{co-NP}$  if there exists a certifier  $C(s, t)$  s.t.

- Input string  $s$  is a **NO** instance if and only if **there exists** a short certificate  $t$  such that  $C(s, t) = \text{YES}$ .
- Alternatively, input string  $s$  is a **YES** instance if and only if **for all** short  $t$ ,  $C(s, t) = \text{NO}$ .

# Co-NP Certifiers and Certificates

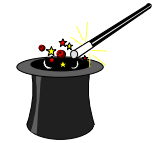
**PRIME:** Given integer  $s$ , is  $s$  prime?

**Observation.**  $s$  is composite if and only if there exists an integer  $1 < t < s$  such that  $s$  is a multiple of  $t$ .

- NO instance:  $s = 437,669$ .

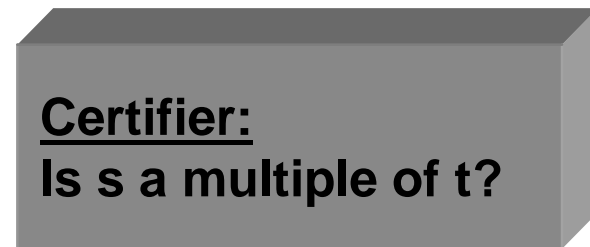
**Conclusions.**

- $\text{PRIME} \in \text{co-NP}$ .
- $\text{COMPOSITE} \in \text{P}$ .



Input s:  
437,669

Certificate t:  
541



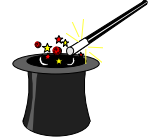
YES

NO

$s$  is a NO instance

no conclusion

# Co-NP Verifiers and Certificates



**COMPOSITE:** Given integer  $s$ , is  $s$  composite?

**Fact (Pratt).**  $s$  is prime if and only if  $s > 2$  is odd and there exists an integer  $1 < t < s$  s.t.

$$t^{s-1} \equiv 1 \pmod{s}$$

$$t^{(s-1)/p} \not\equiv 1 \pmod{s}$$

for all prime divisors  $p$  of  $s-1$

- **NO instance:** 437,677.

**Conclusions.**

- **COMPOSITE**  $\in$  co-NP.
- **PRIME**  $\in$  NP.

Input  $s$ :  
437,677

Certificate  $t$ :  
17

Certifier:

$$17^{(s-1)} \bmod s = 1$$

$$s - 1 = 2 \times 2 \times 3 \times 36,473$$

$$17^{(s-1)/2} \bmod s = 437,676$$

$$17^{(s-1)/3} \bmod s = 329,415$$

$$17^{(s-1)/36,473} \bmod s = 305,452$$

YES

NO

$s$  is a NO instance

no conclusion

# NP = co-NP ?

**Fundamental question: does NP = co-NP?**

- Do YES-instances have short certificates if and only if NO-instances have short certificates.
- Consensus opinion: no.

**Theorem.** If  $NP \neq co-NP$ , then  $P \neq NP$ .

**Proof.** We prove if  $P = NP$ , then  $NP = co-NP$ .

- Key idea:  $P$  is closed under complementation, so if  $P = NP$ , then  $NP$  is closed under complementation as well.
- More formally, using the assumption  $P = NP$ :

$$X \in NP \Rightarrow X \in P \Rightarrow \bar{X} \in P \Rightarrow \bar{X} \in NP \Rightarrow X \in co-NP$$

$$X \in co-NP \Rightarrow \bar{X} \in NP \Rightarrow \bar{X} \in P \Rightarrow X \in P \Rightarrow X \in NP$$

- Thus,  $NP \subseteq co-NP$  and  $co-NP \subseteq NP$ , so  $co-NP = NP$ .

# Good Characterizations

**Good characterization:**  $NP \cap co-NP$ .

- If problem  $X$  is in  $NP$  and  $co-NP$ , then:
  - for YES instance, there is a short certificate
  - for NO instance, there is a short certificate
- Provides conceptual leverage for reasoning about a problem.

**Examples.**

- **MAX-FLOW:** given a network, is there an  $s$ - $t$  flow of value  $\geq W$ .
  - if yes, can exhibit  $s$ - $t$  flow that achieves this value
  - if no, can exhibit  $s$ - $t$  cut whose capacity is less than  $W$
- **PERFECT-MATCHING:** given a bipartite graph, is there a perfect matching.
  - if yes, can exhibit a perfect matching
  - if no, can exhibit a set of vertices  $S \subseteq L$  such that the total number of neighbors of  $S$  is strictly less than  $|S|$



# Good Characterizations

**Observation.**  $P \subseteq NP \cap \text{co-NP}$ .

- Proof of max-flow min-cut theorem and Hall's theorem led to stronger results that max-flow and bipartite matching are in P.
- Sometimes finding a good characterization seems easier than finding an efficient algorithm: linear programming.

**Fundamental question:** does  $P = NP \cap \text{co-NP}$ ?

- Mixed opinions.
- Many examples where problem found to have a non-trivial good characterization, but only years later discovered to be in P.
- Note:  $\text{PRIME} \in NP \cap \text{co-NP}$ , but not known to be in P.

# A Note on Terminology

**Knuth. (SIGACT News 6, January 1974, p. 12 – 18)**

**Find an adjective  $x$  that sounds good in sentences like.**

- **FIND-TSP-TOUR is  $x$ .**
- **It is  $x$  to decide whether a given graph has a Hamiltonian cycle.**
- **It is unknown whether FACTOR is an  $x$  problem.**

**Note:  $x$  does not necessarily imply that a problem is in NP or even a decision problem.**

**Knuth's original suggestions.**

- **Hard.**
- **Tough.**
- **Herculean.**
- **Formidable.**
- **Arduous.**

# A Note on Terminology

## Some English word write-ins.

- **Impractical.**
- **Bad.**
- **Heavy.**
- **Tricky.**
- **Intricate.**
- **Prodigious.**
- **Difficult.**
- **Intractable.**
- **Costly.**
- **Obdurate.**
- **Obstinate.**
- **Exorbitant.**
- **Interminable.**

# A Note on Terminology

## Hard-boiled. (Ken Steiglitz)

- In honor of Cook.

## Hard-ass. (Al Meyer)

- Hard as satisfiability.

## Sisyphean. (Bob Floyd)

- Problem of Sisyphus was time-consuming.
- Hercules needed great strength.
- Problem: Sisyphus never finished his task  $\Rightarrow$  unsolvable.

## Ulyssean. (Don Knuth)

- Ulysses was noted for his persistence and also finished.

# A Note on Terminology: Made-Up Words

## Exparent. (Mike Paterson)

- exponential + apparent

## Perarduous. (Mike Paterson)

- through, in space or time + completely

## Supersat. (Al Meyer)

- greater than or equal to satisfiability

## Polychronious. (Ed Reingold)

- enduringly long; chronic
- Appears in Webster's 2<sup>nd</sup> unabridged, but apparently in no other dictionary.

# A Note on Terminology: Acronyms

## **PET.** (Shen Lin)

- Probably exponential time.
- Provably exponential time, previously exponential time.

## **GNP.** (Al Meyer)

- Greater than or equal to NP in difficulty.
- Costing more than GNP to resolve.

# A Note on Terminology: Consensus

## NP-complete.

- A problem in NP such that every other problem in NP transforms to it.

## NP-hard. (Bell Labs, Steve Cook, Ron Rivest, Sartaj Sahni)

- A problem such that every problem in NP transforms to it.

## Knuth's conclusion.

- "create research workers are as full of ideas for new terminology as they are empty of enthusiasm for adopting it."

# Summary

**Many fundamental problems are NP-complete.**

- **TSP, 3-CNF-SAT, 3-COLOR, CLIQUE, . . . .**

**Theory says we probably won't be able to design efficient algorithms for NP-complete problems.**

- **You will likely run into these problems in your scientific life.**
- **If you know about NP-completeness, you can identify them and avoid wasting time.**