Complexity: P vs NP

P and NP

- We discussed big-Oh notation in the last couple lectures
  - Big-Oh notation can be used to compare the speed of algorithms, some are just not feasible for computing (e.g., $O(2^n)$)
  - Similar notion can also be applied to space (the amount of memory needed for computation)
  - We can generalize the notion of complexity (mostly in speed or time) in the phrase of "P vs. NP"

What does P and NP mean?

- If a problem can be solved in $O(n^k)$ where k is a fixed constant (note k could be zero!), we say this type of problems belong to the class of **P** (for Polynomial).
- If a problem can be solved in $O(k^n)$ where k is a fixed constant (k > 1) we say the time complexity is **exponential** and it is not practical to solve problems in this manner.
- However, is it **possible** that solutions of polynomial time exist for a problem that is known can be solved in exponential time?

NP: non-deterministic polynomial

- In particular, if we can verify a solution in polynomial time, does a polynomial time solution exists?
- The class of problems (examples follow) that can be verified in polynomial time is called **NP** problems (non-deterministic polynomial problems).
- An exponential ($O(2^n)$) solution to these problems often exist. We are searching for $O(n^k)$ solutions.

Examples of **NP** Problems

- **Subset Sum Problem** (Wikipedia): Given a set of integers, does some nonempty subset of them sum to 0?
  - For instance, does a subset of the set {−2, −3, 15, 14, 7, −10} add up to 0?
  - The answer "yes, because the subset {−2, −3, −10, 15} adds up to zero" can be quickly verified with three additions.
  - There is no known algorithm to find such a subset in polynomial time. There is one, however, in exponential time, which consists of 2^n-n-1 tries.

Examples of **NP** Problems

- **Traveling Salesperson Problem**
  - Given a collection of cities, can we find a route such that the salesperson can start from an originating city, visit every other city exactly once, and come back to the starting place?

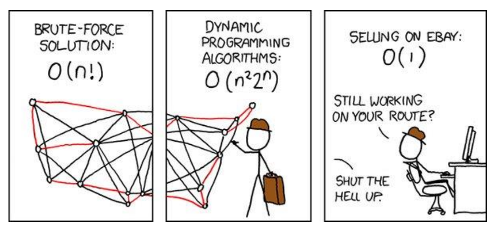
Which route should you take?

Which route should you take?

Which route should you take?

Traveling Salesman Problem:

O(n!) which is worse than O(2^n)



How long does it take us to solve a problem?

?

Slow Fast

Will some algorithms EVER be faster?

How long does it take us to solve a problem?

?

Slow Fast

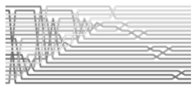
~~**Will some algorithms EVER be faster?**~~

Will some questions EVER be answered?

Provably Slow — **Provably Fast**

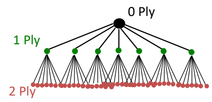
Slow to solve
Slow to verify

5127
x 4265
25635
307620
1025400
20508000
21866655



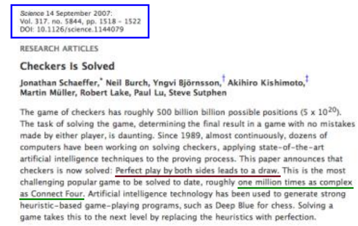
Fast to solve
Fast to verify

Why Games are Slow



Branching Factor Estimates
for different two-player games

Tic-tac-toe	4
Connect Four	7
Checkers	10
Othello	30
Chess	40
Go	300



Science 14 September 2007:
Vol. 317. no. 5844, pp. 1518 - 1522
DOI: 10.1126/science.1144079

RESEARCH ARTICLES

Checkers Is Solved

Jonathan Schaeffer,* Neil Burch, Yngvi Björnsson,† Akihiro Kishimoto,‡ Martin Müller, Robert Lake, Paul Lu, Steve Sutphen

The game of checkers has roughly 500 billion billion possible positions (5×10^{20}). The task of solving the game, determining the final result in a game with no mistakes made by either player, is daunting. Since 1989, almost continuously, dozens of computers have been working on solving checkers, applying state-of-the-art artificial intelligence techniques to the proving process. This paper announces that checkers is now solved: Perfect play by both sides leads to a draw. This is the most challenging popular game to be solved to date, roughly one million times as complex as Connect Four. Artificial intelligence technology has been used to generate strong heuristic-based game-playing programs, such as Deep Blue for chess. Solving a game takes this to the next level by replacing the heuristics with perfection.

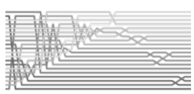
Checkers was solved in 2007. Other partially solved games include “Go” and “Chess.”

Provably Slow — **?** — **Provably Fast**

Slow to solve
Slow to verify

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

5127
x 4265
25635
307620
1025400
20508000
21866655



Fast to solve
Fast to verify

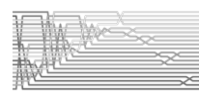
Provably Slow — **?** — **Provably Fast**

Slow to solve
Slow to verify

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

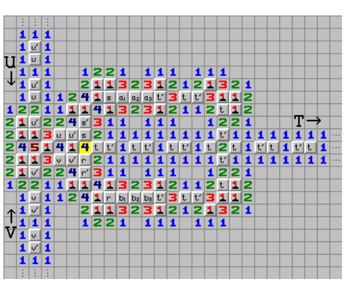
?
Fast to verify

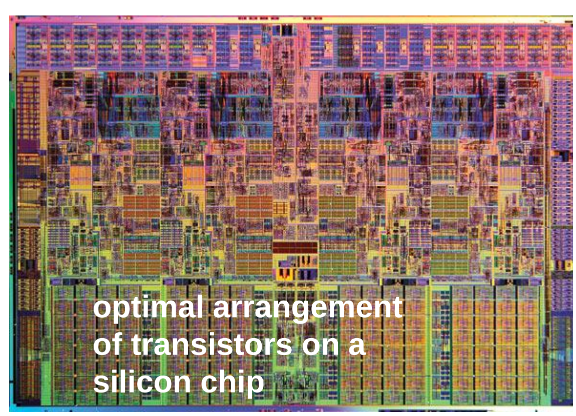
5127
x 4265
25635
307620
1025400
20508000
21866655



Fast to solve
Fast to verify

How do you play the perfect game of Minesweeper?

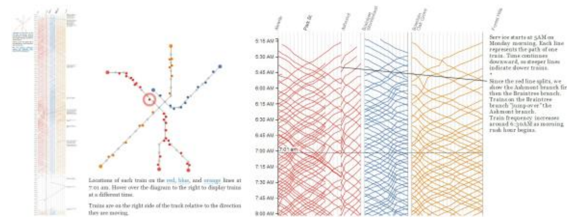




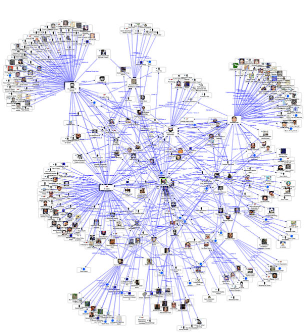
How can we schedule exams so that no one has 3 in one day?

FINAL EXAM DATE AND TIME	WEDNESDAY MAY 1, 2019	THURSDAY MAY 2, 2019	FRIDAY MAY 3, 2019	MONDAY MAY 6, 2019	TUESDAY, MAY 7, 2019	WEDNESDAY MAY 8, 2019
8:00 – 11:00	ECON 313 ENGR 212 MATH 216, 240/241 MECH 202, 216, 392	TR 8 a.m.	TR 1 p.m. T 1 p.m. R 1 p.m.	CSCI 208 ECON 280 MATH 202, 211, 212, 222 MECH 302 POLS 296	TR 9:30 a.m.	MWF 12 p.m. TR 11 a.m.
11:45 – 2:45	MWF 9 a.m. MW 8:30 a.m. MF 8:30 a.m. WF 8:30 a.m.	CEEG 242 CSCI 206 ECON 103, 259 ENGR 214 MECH 312	MWF 11 a.m.	MWF 1 p.m.	CSCI 203 ECON 257 ENGR 101 POLS 170-02/03	MWF 4 p.m. TR 4 p.m.
3:30 – 6:30	TR 2:30 p.m.	MWF 3 p.m. MW 3 p.m. WF 3 p.m. MF 3 p.m.	CEEG 330 ECON 258 GEOL 203 PHYS 212	MWF 10 a.m. MGMT 101	MWF 8 a.m.	BIOL 208 CSCI 205 GEOL 204 MATH 245
7:30 – 10:30	R 7 p.m. WR 7 p.m.	W 7 p.m. MW 7 p.m. MECH 220, 353	No Exams	M 7 p.m. MR 7 p.m.	MWF 2 p.m.	No Exams

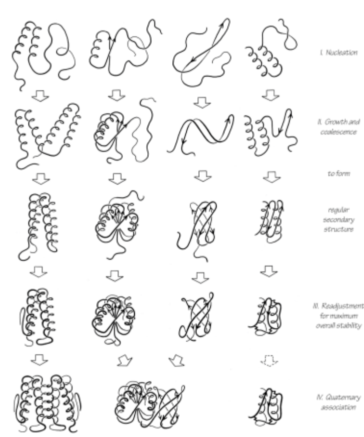
How can we optimize the schedule for subways and buses?



What's the best place to put your manufacturing plant?



What is the largest group of your friends that all know each other?



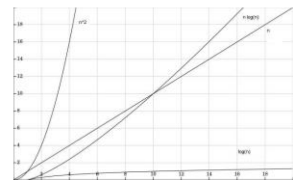
How do you fold a protein?

P = NP in one question:

Does being able to **quickly verify** correct answers mean that there is also *some* way to **quickly find the answers**?

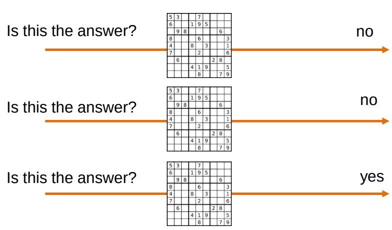
Some Things to Know

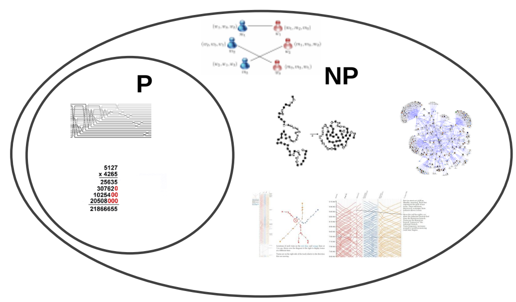
1) When we talk how long it takes to answer question we're talking about *all potential* versions of that question.
   Scaling **Worst Case**
2) P stands for **Polynomial Time**

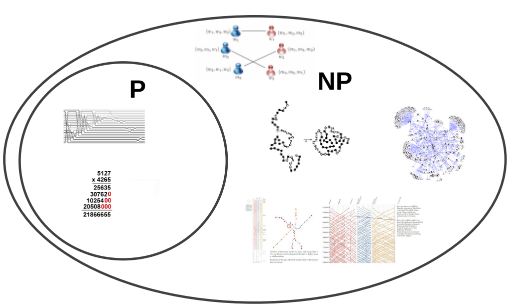


3) NP is **Nondeterministic Polynomial Time**

Since checking answers is easy, if you could check every possible answer simultaneously, you could figure out the true answer pretty quickly

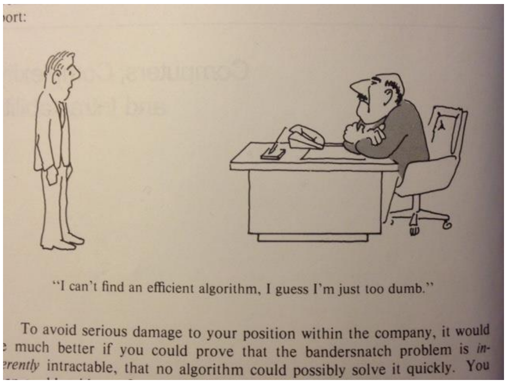






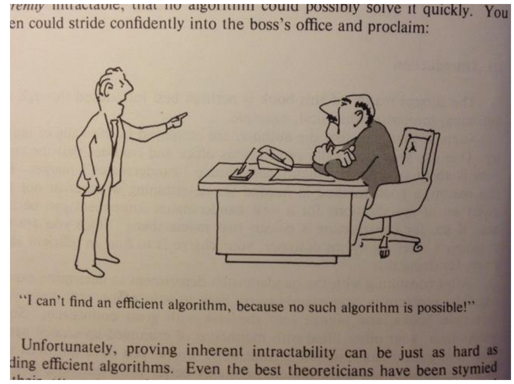
How to tackle this problem (P == NP?)

- Scientists try to identify a set of problems that are at least as hard (**NP Hard Problems**).
- When a new, unknown problem is encountered, if one can deduce the new problem into one of the NP-Hard Problems, then we know the nature of the new problem.


ort:

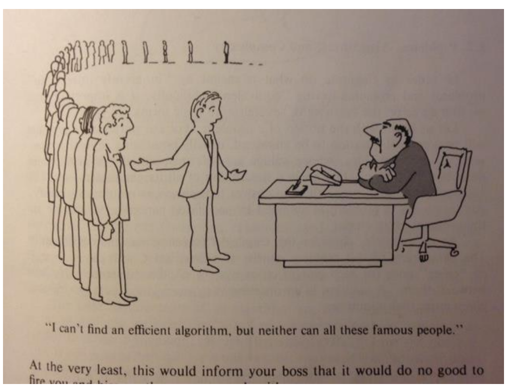
"I can't find an efficient algorithm, I guess I'm just too dumb."

To avoid serious damage to your position within the company, it would e much better if you could prove that the bandersnatch problem is *in- erently* intractable, that no algorithm could possibly solve it quickly. You


rently intractable, that no algorithm could possibly solve it quickly. You en could stride confidently into the boss's office and proclaim:

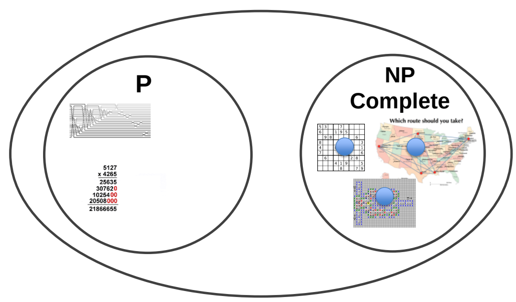
"I can't find an efficient algorithm, because no such algorithm is possible!"

Unfortunately, proving inherent intractability can be just as hard as ding efficient algorithms. Even the best theoreticians have been stymied

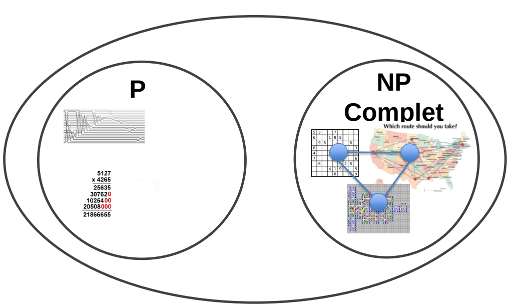

"I can't find an efficient algorithm, but neither can all these famous people."

At the very least, this would inform your boss that it would do no good to

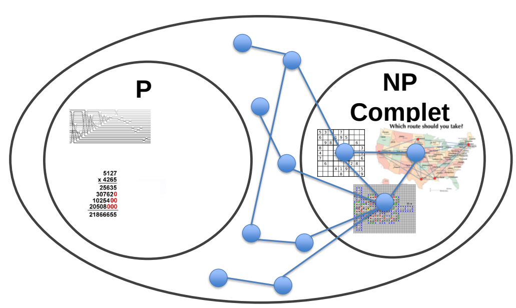
So.. does P = NP?



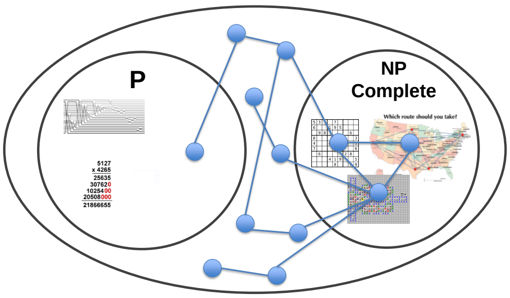
So.. does P = NP?



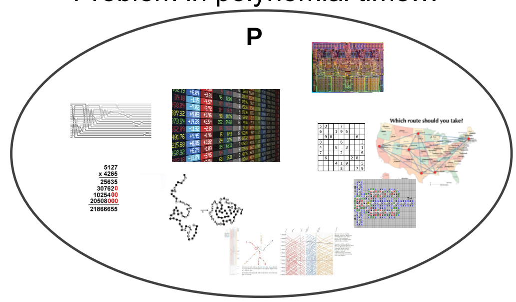
So.. does P = NP?



So.. does P = NP?



If we just solve one NP-Complete Problem in polynomial time…

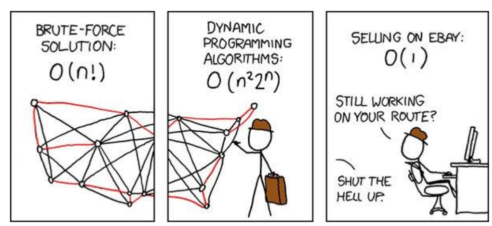


Scott Aaronson, MIT

The Philosophical Argument

If P = NP, then the world would be a profoundly different place than we usually assume it to be. There would be no special value in “creative leaps,” no fundamental gap between solving a problem and recognizing the solution once it’s found… if this is the sort of universe we inhabited, why wouldn’t we already have evolved to take advantage of it?

Ending on some good(ish) news



One More Problem to Solve

Choose **any positive integer** for a, b, and c
Choose **any integer > 2** for n

Find an instance in which this is true:

$$a^n + b^n = c^n$$

```python
while True:
    # Slowly increase a, b, c, n
    # Testing each combination along the way
    if check(a, b, c, n)
        break


def check(a, b, c, n):
    if a**n + b**n == c**n:
        return True
    else:
        return False
```

Fermat's Last Theorem

```python
while True:
    # Slowly increase a, b, c, n
    # Testing each combination along the way
    if check(a, b, c, n)
        break


def check(a, b, c, n):
    if a**n + b**n == c**n:
        return True
    else:
        return False
```

When will this program stop?