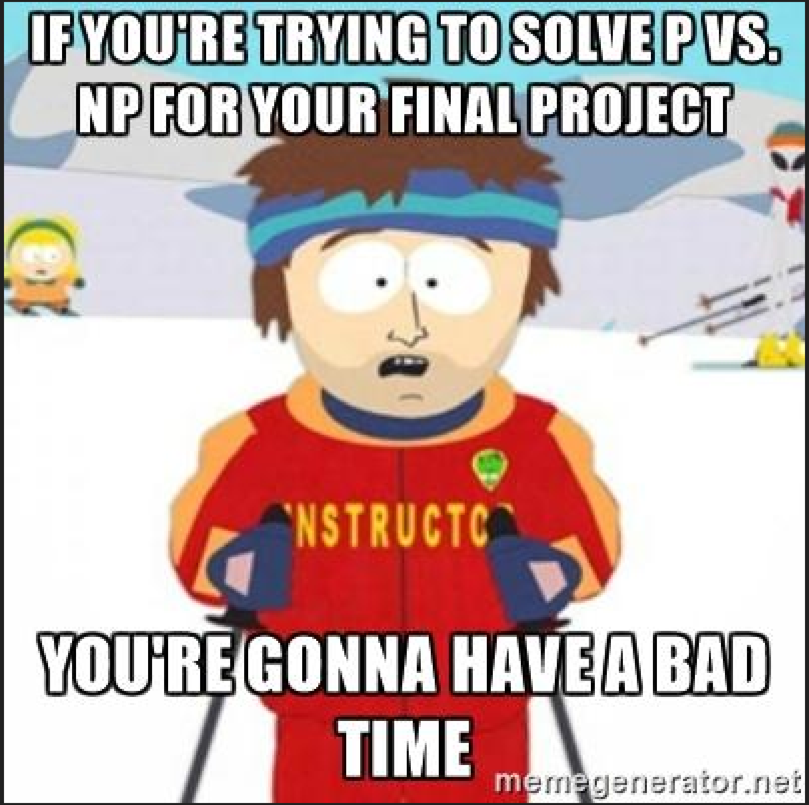
P vs. NP

The Greatest Unsolved Problem in Computer Science

And perhaps all of Mathematics!

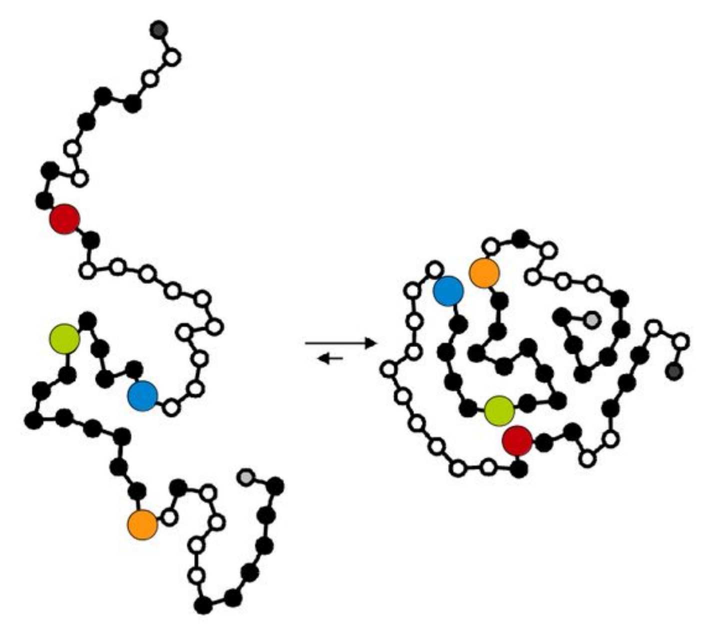
P=NP
$e^{\pi i} = -1$

EMERGENCY BEANS
CONDENSED MILT
P
NP


IF YOU'RE TRYING TO SOLVE P VS. NP FOR YOUR FINAL PROJECT
NSTRUCTO
YOU'RE GONNA HAVE A BAD TIME
memegenerator.net

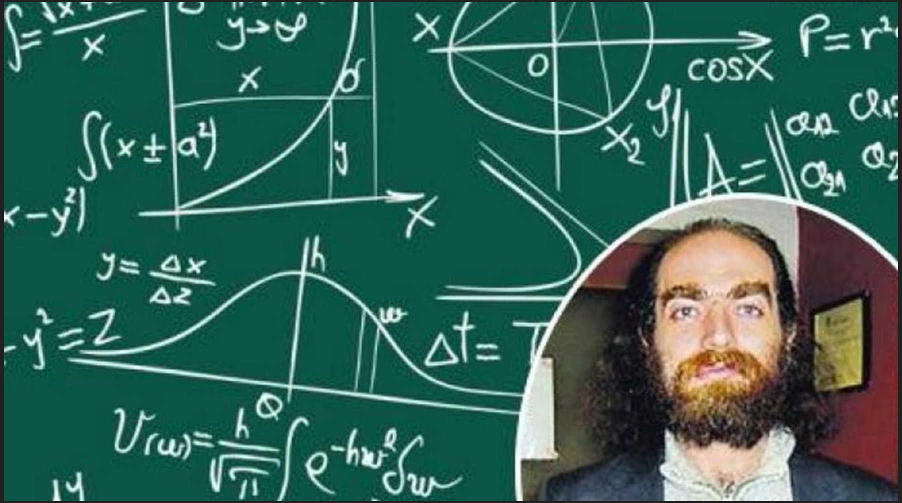
Why care about Theory?

- Algorithms matter, so we should study them!
- We can know the limits of computation
  - Models of Computers - “What is a Computer?”
  - Models of algorithms - “In what ways are algorithms similar to one another?”
- Helps solve problems and use solutions to solve other problems



Motivation

- Are algorithms invented or discovered?
- Is there no ‘fast’ algorithm to solve an sudoku? Or are we just too dumb to discover it?
- Literally a ‘Million Dollar Question’



CS50

P vs. NP

=

Are problems that are easy to check also easy to solve?

History (<1965)

Slow

- Factoring
- Traveling Salesman Problem
- Determine if number is prime
- Discrete Fourier Transform

Fast

- Greatest Common Divisor
- Sorting

History (1965)

Slow

- Factoring
- Traveling Salesman Problem
- Determine if number is prime

Fast

- Greatest Common Divisor
- Sorting
- Discrete Fourier Transform

Cooley Tukey FFT!

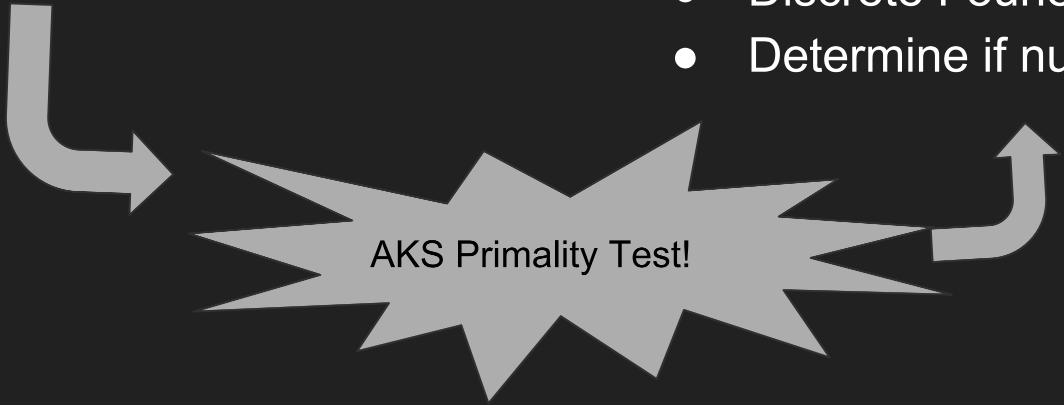
History (2002)

Slow

- Factoring
- Traveling Salesman Problem

Fast

- Greatest Common Divisor
- Sorting
- Discrete Fourier Transform
- Determine if number is prime



AKS Primality Test!

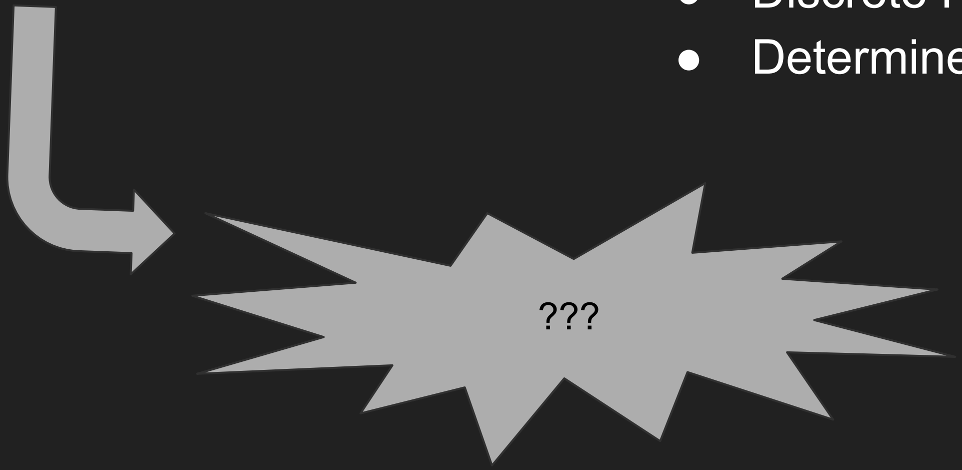
History

Slow

- Factoring
- Traveling Salesman Problem

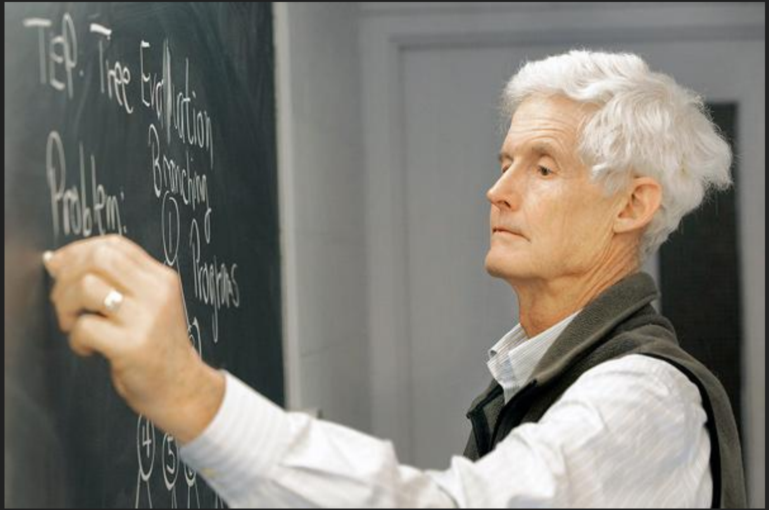
Fast

- Greatest Common Divisor
- Sorting
- Discrete Fourier Transform
- Determine if number is prime



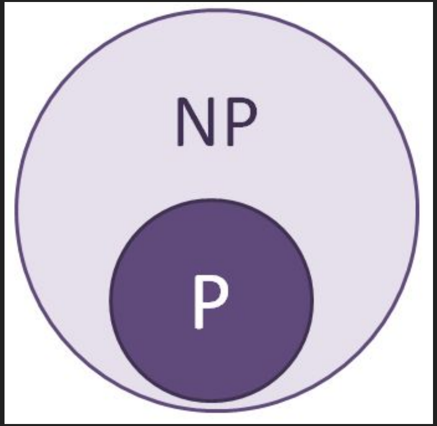
History

- Stephen Cook, Leonid Levin - Cook-Levin theorem
  - Found that some algorithmic problems are connected by core difficulty (NP-Completeness)
  - What does this mean?



What is P? What is NP? Why must they fight?

- P and NP are sets of problems that require an algorithm to solve
- P vs. NP is really the question: is P = NP ?
  - We know that P⊆NP



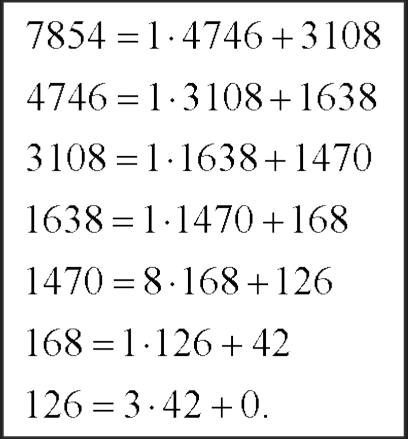
The Set: P (Polynomial-Time)

The Set P is the set of problems for which there exists a polynomial time algorithm that generates a solution (Algorithm is O(n^k); n is size of problem)

- Basically: Problems that can be **solved** quickly.

Problems include:

- Finding GCD
- Linear Programming
- Determining if a number is prime*
- Multiplication



$$7854 = 1 \cdot 4746 + 3108$$

$$4746 = 1 \cdot 3108 + 1638$$

$$3108 = 1 \cdot 1638 + 1470$$

$$1638 = 1 \cdot 1470 + 168$$

$$1470 = 8 \cdot 168 + 126$$

$$168 = 1 \cdot 126 + 42$$

$$126 = 3 \cdot 42 + 0.$$

*=Not obvious! Took smart people until 2002

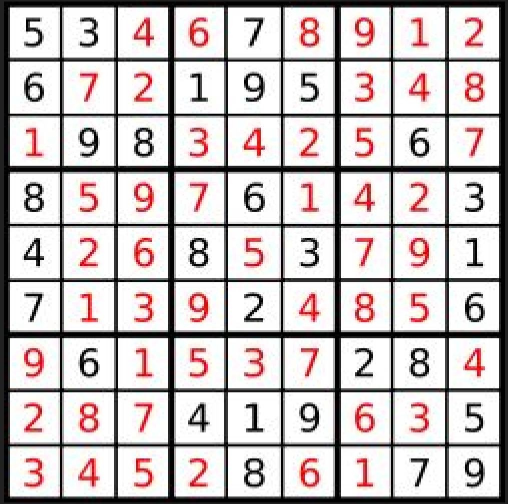
The Set: NP (Nondeterministic Polynomial-Time)

The Set NP is the set of decision problems for which there exists a polynomial time algorithm to check if a solution is correct

- Basically: Problems that can be **checked** quickly.

Problems include:

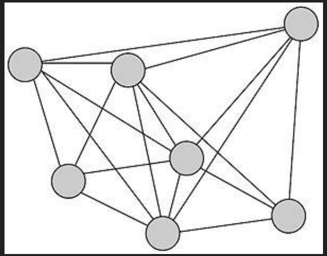
- Sudoku
- Factoring
- Traveling Salesman Problem: inputs {V}, {E = VxV}, k
- Multiplication

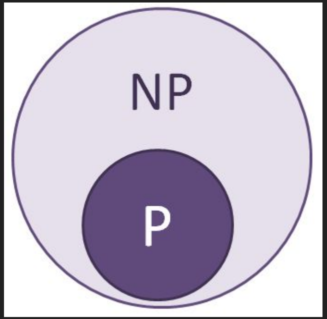


5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

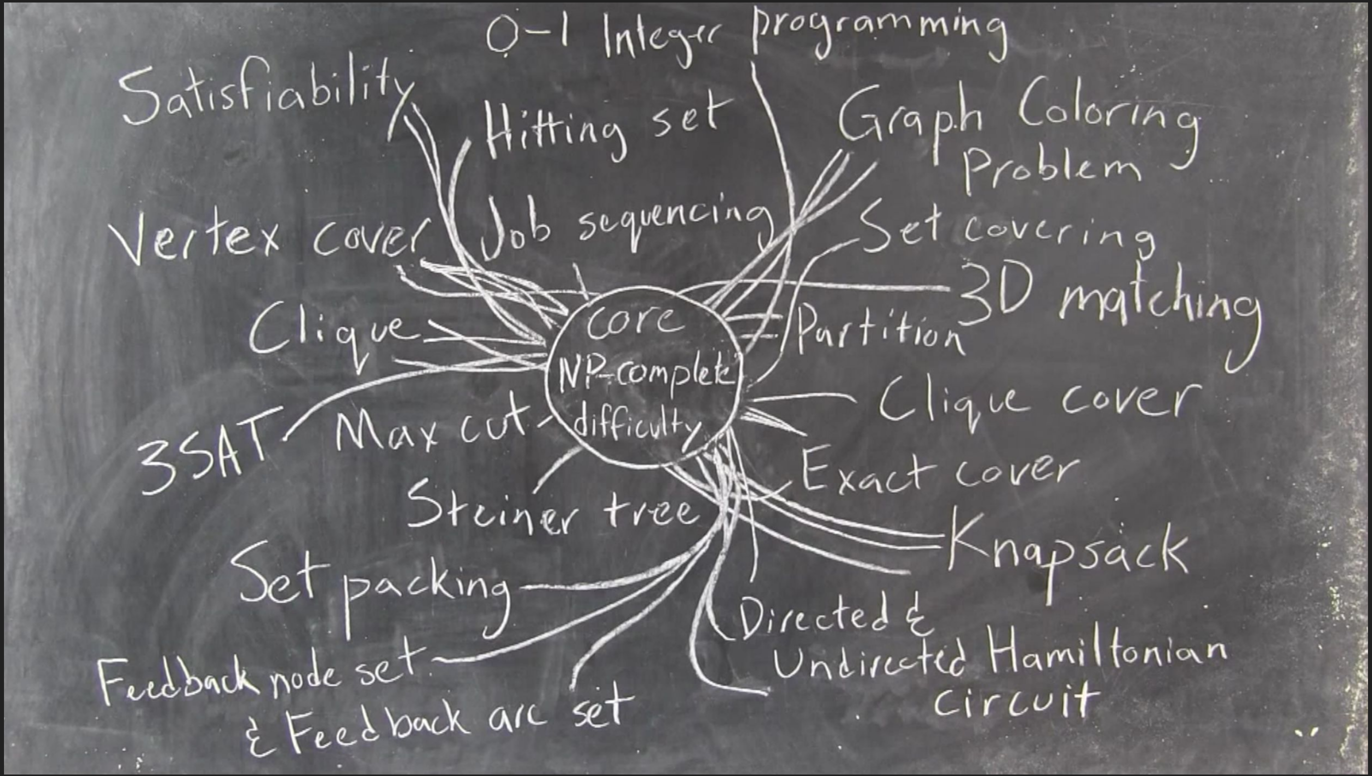
At least in NP? Or also in P?

- Sorting a list?
- Multiplication?
- Given sets of Vertices (V) and Edges (E = V x V), is the graph <u>connected</u>?
- Rubik’s cube?
- Best move in Chess?
- Subset Sum?
  - Ex: Is there a subset of the set {3, 10, -4, 5, -16, -3} that sums to -1 ?



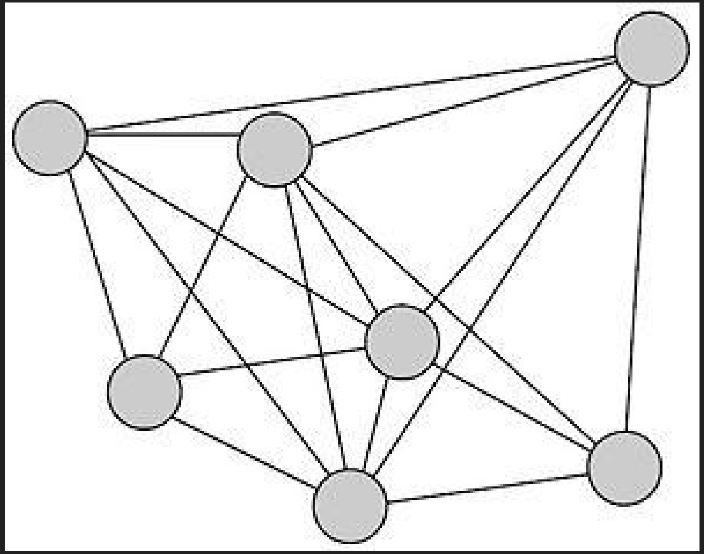


NP-Complete


0-1 Integer programming
Satisfiability
Hitting set
Graph Coloring Problem
Job sequencing
Set covering
Vertex cover
3D matching
core NP-complete difficulty
Clique
Partition
Clique cover
3SAT
Max cut
Exact cover
Steiner tree
Knapsack
Set packing
Directed & Undirected Hamiltonian circuit
Feedback node set & Feedback arc set

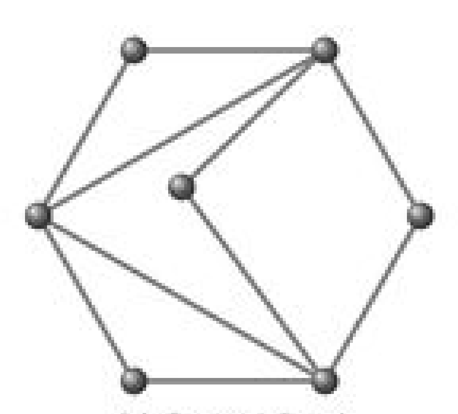
Reduction

- A Reduction is a Polynomial-time algorithm that converts a solution of one problem to a solution of another problem.
  - Independent Set Problem
  - Vertex Cover Problem
  - Clique Problem



Reduction

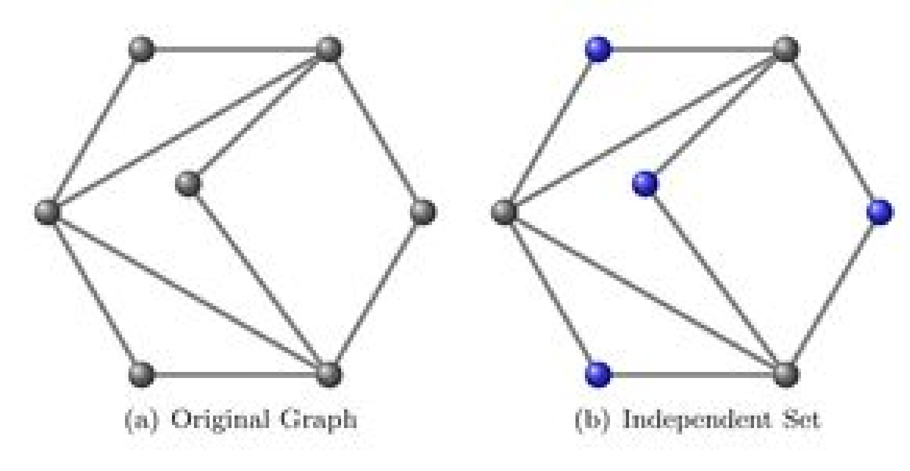
- A Reduction is a Polynomial-time algorithm that converts a solution of one problem to a solution of another problem.



(a) Original Graph

Reduction

- A Reduction is a Polynomial-time algorithm that converts a solution of one problem to a solution of another problem.

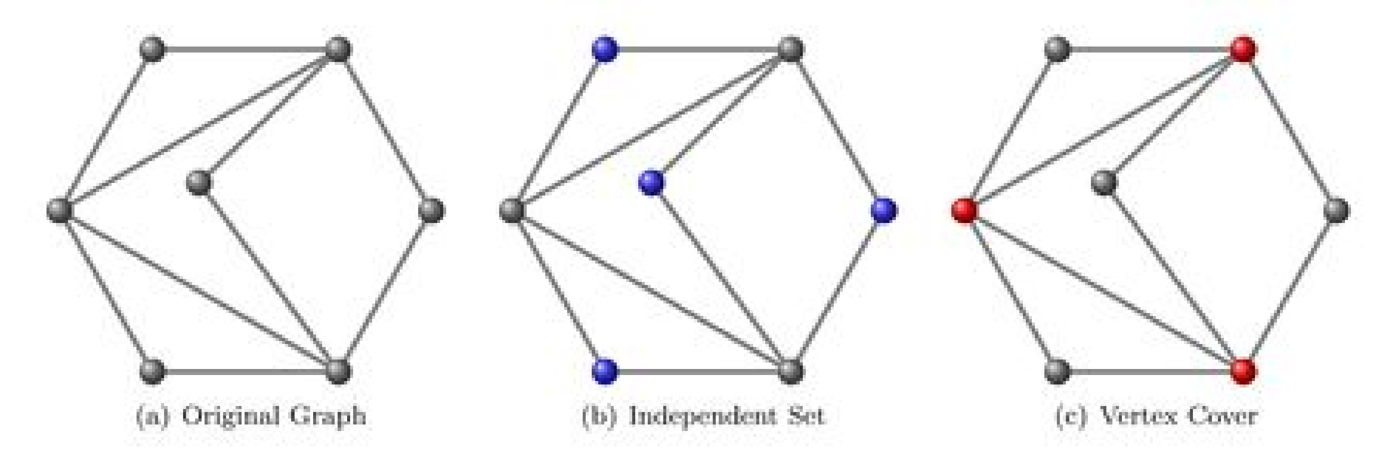


(a) Original Graph

(b) Independent Set

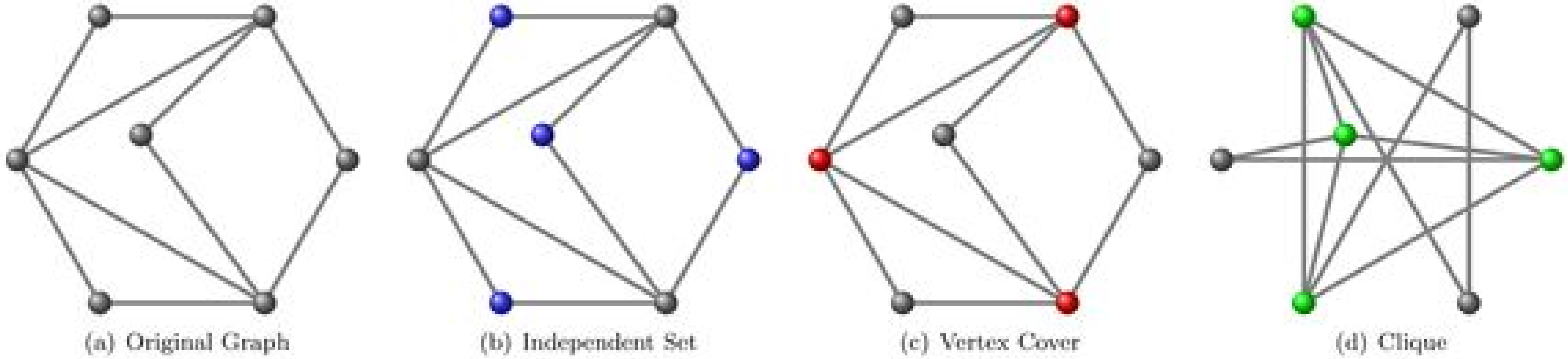
Reduction

- A Reduction is a Polynomial-time algorithm that converts a solution of one problem to a solution of another problem.



Reduction

- A Reduction is a Polynomial-time algorithm that converts a solution of one problem to a solution of another problem.


(a) Original Graph

(b) Independent Set

(c) Vertex Cover

(d) Clique

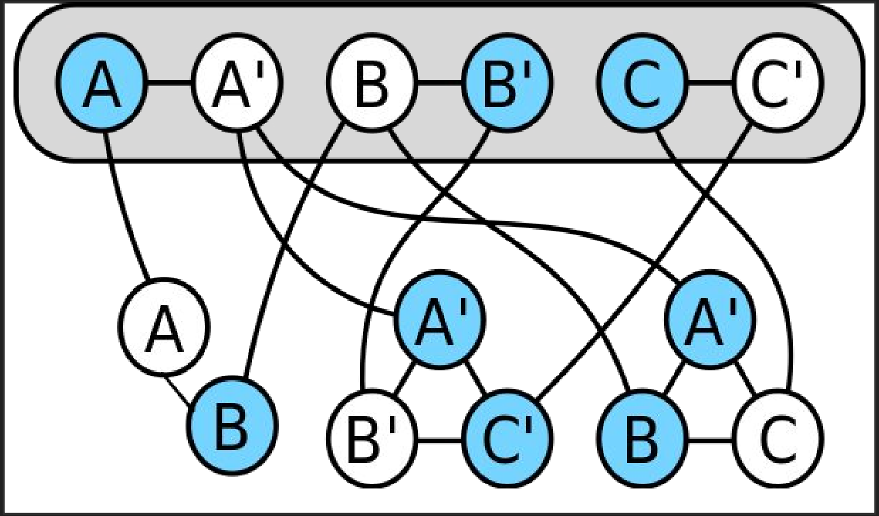
Figure 1: Relations among Independent Set, Vertex Cover, and Clique

Reduction

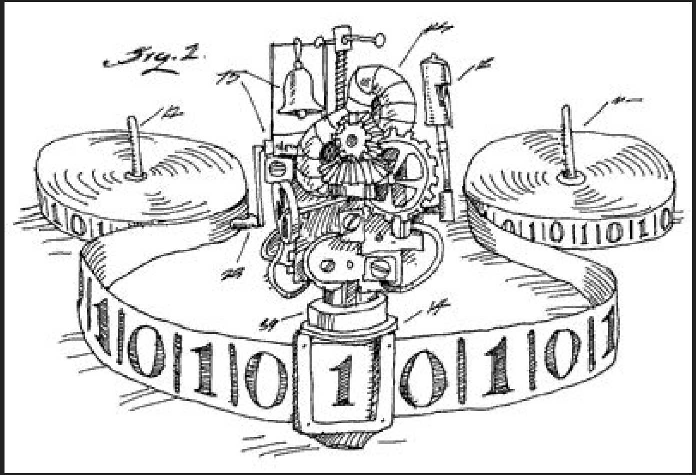
- A Reduction is a Polynomial-time algorithm that converts a solution of one problem to a solution of another problem.

$(A \lor B) \land (\neg A \lor \neg B \lor \neg C) \land (\neg A \lor B \lor C)$

SAT
->
Vertex Cover

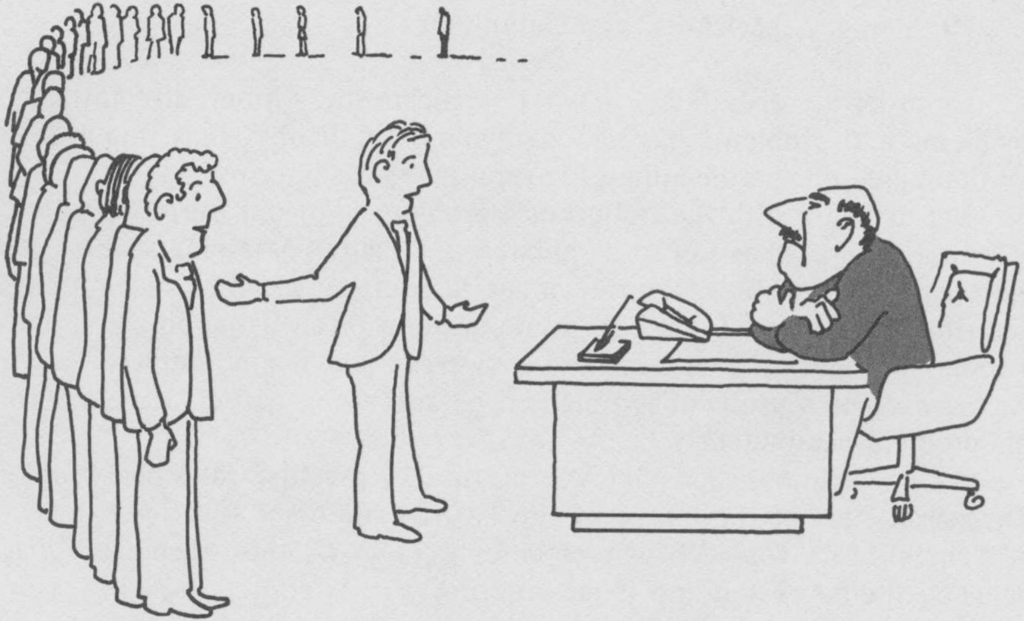


Turing Machine



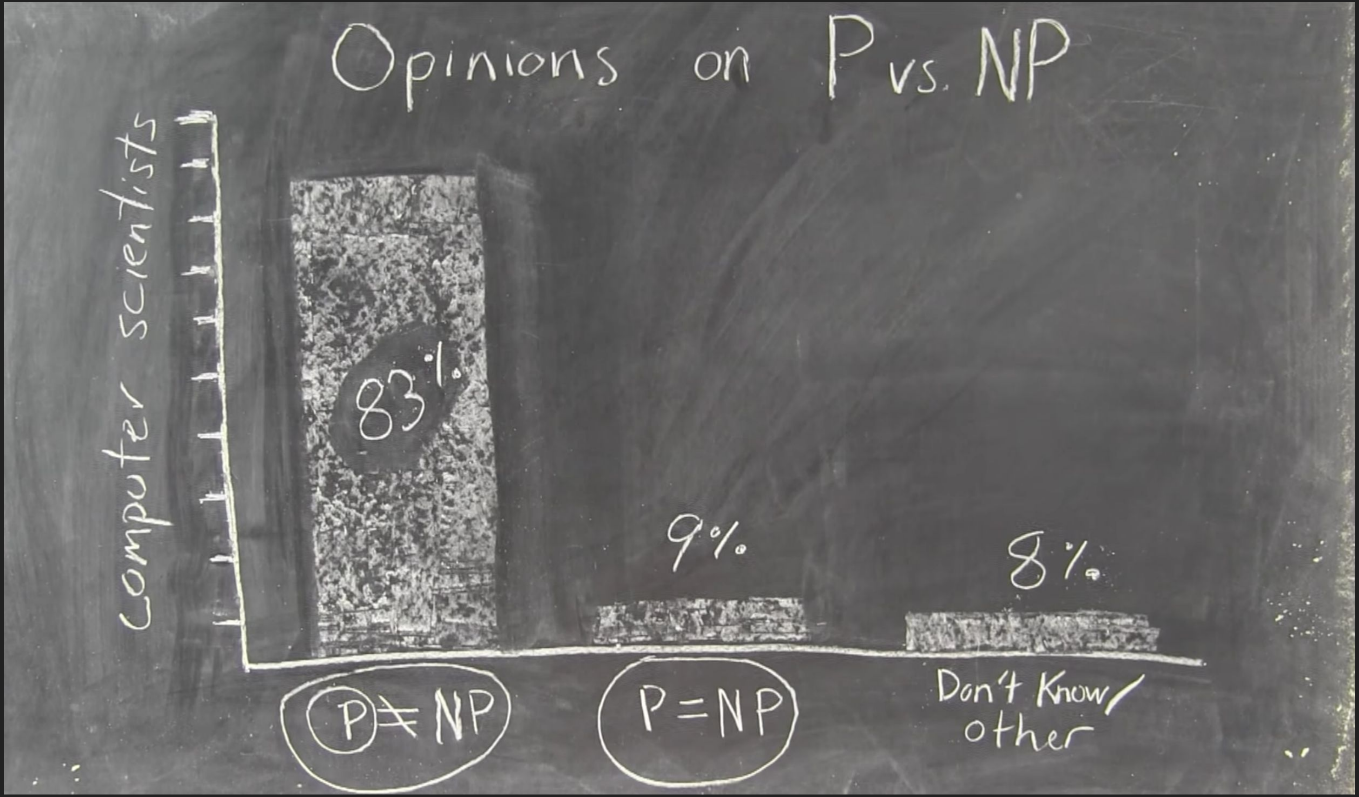
- Theoretical Model of a computer
- Alan Turing - ‘Mathematical functions on numbers can be just as well executed by a Turing Machine’
- A Reduction of a Computer to its simplest abilities:
  - Ability to read from memory, ability to write to memory.
  - Given some input memory, run an algorithm on the Turing Machine (TM)

What do the Smart People Think? Read: Not Sammy



“I can’t find an efficient algorithm, but neither can all these famous people.”

P ≠ NP (Probably)



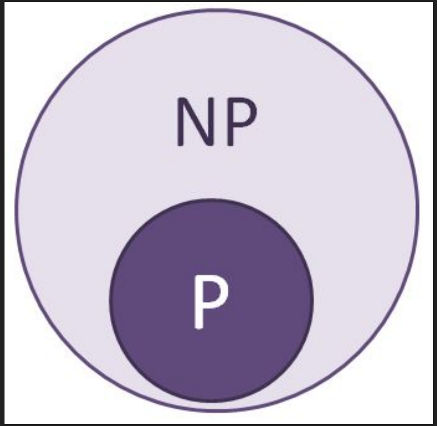
How to Prove it

To prove P = NP:

- Give a Polynomial time algorithm to solve **ANY** NP-Complete problem

To prove P ≠ NP:

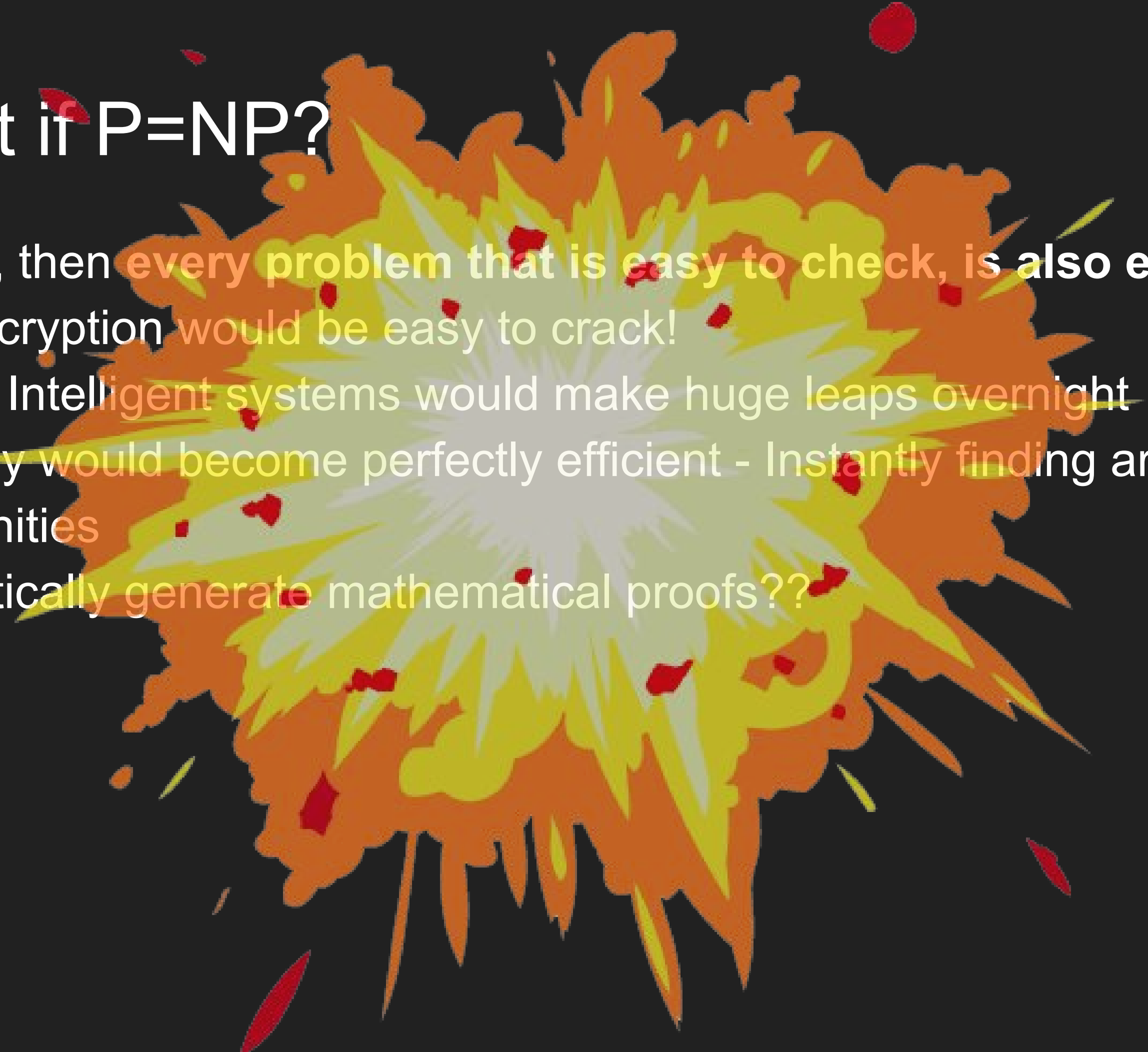
- Prove that there exists **NO ALGORITHM** to solve some NP problem in polynomial time
  - This is not an easy task!



So What if P=NP?

- If P=NP, then **every problem that is easy to check, is also easy to solve.**
- RSA Encryption would be easy to crack!
- Artificial Intelligent systems would make huge leaps overnight
- Economy would become perfectly efficient - Instantly finding arbitrage opportunities
- Automatically generate mathematical proofs??

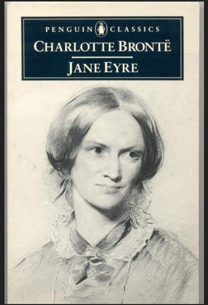
So What if P=NP?



- If P=NP, then **every problem that is easy to check, is also easy to solve.**
- RSA Encryption would be easy to crack!
- Artificial Intelligent systems would make huge leaps overnight
- Economy would become perfectly efficient - Instantly finding arbitrage opportunities
- Automatically generate mathematical proofs??

Philosophy

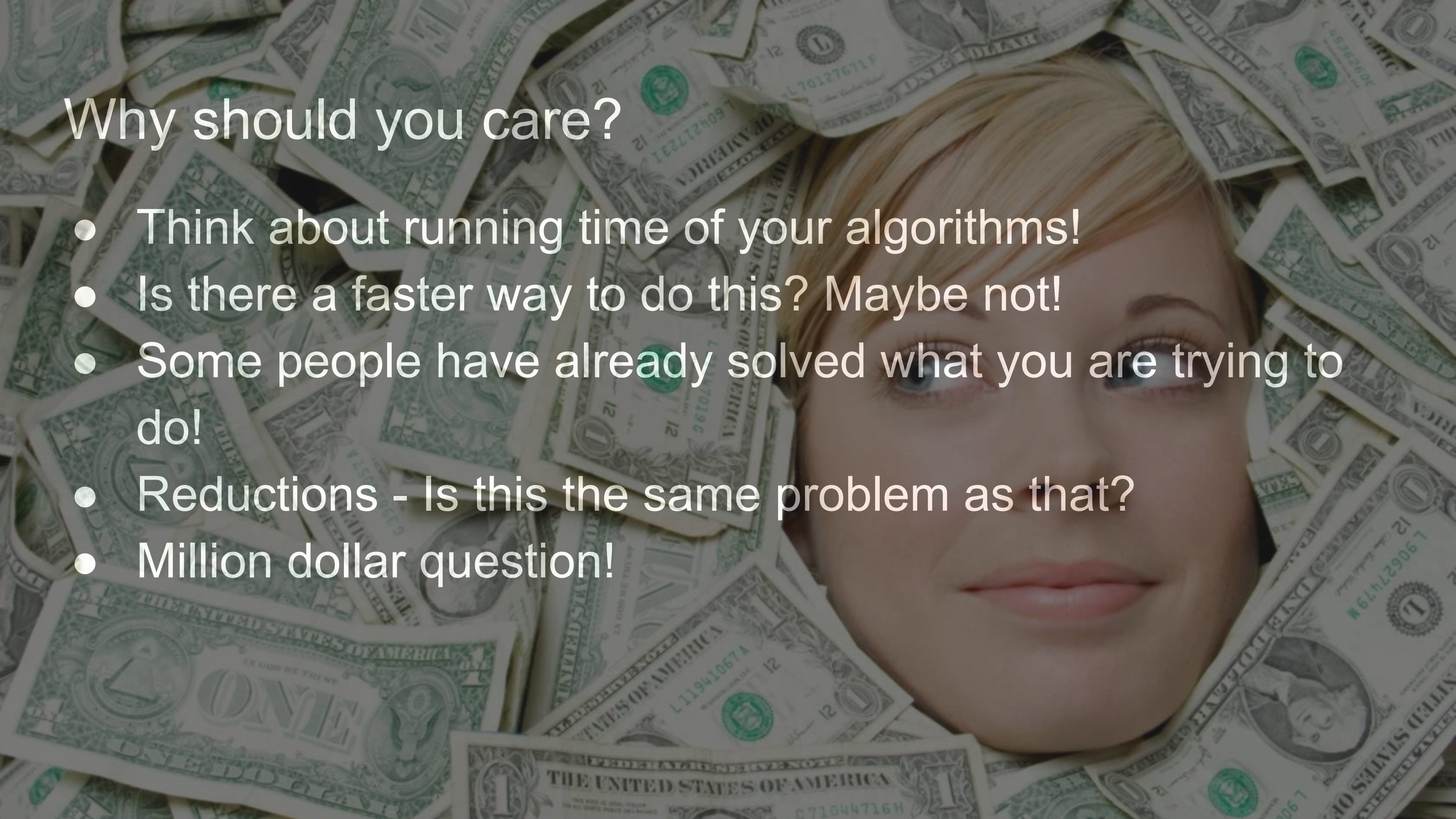
- Proving Things?
- Comedy?
- Music?
- Art?
- Literature?



“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. Everyone who could appreciate a symphony would be Mozart; everyone who could follow a step-by-step argument would be Gauss” - Scott Aaronson, MIT

Why should you care?

- Think about running time of your algorithms!
- Is there a faster way to do this? Maybe not!
- Some people have already solved what you are trying to do!
- Reductions - Is this the same problem as that?
- Million dollar question!



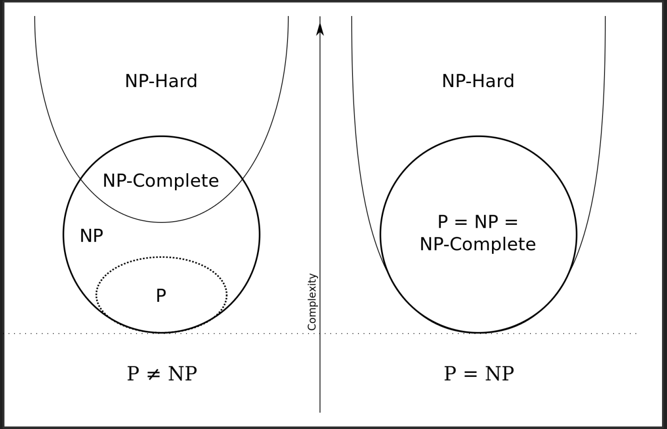
Why should you care?

- Think about running time of your algorithms!
- Is there a faster way to do this? Maybe not!
- Some people have already solved what you are trying to do!
- Reductions - Is this the same problem as that?
- Million dollar question!

Thank You!

P vs. NP Page:
https://www.win.tue.nl/~gwoegi/P-versus-NP.htm

P vs. NP and the Complexity Zoo:
https://www.youtube.com/watch?v=YX40hbAHx3s


NP-Hard
NP-Complete
NP
P
P ≠ NP
Complexity
NP-Hard
P = NP =
NP-Complete
P = NP