

CSE 332 Winter 2026

Lecture 25: P and NP

Nathan Brunelle

<http://www.cs.uw.edu/332>

Studying Complexity and Tractability

- Organizing problems into complexity classes helps us to reason more carefully and flexibly about tractability
- The goal for each problem is to either
 - Find an efficient algorithm if it exists
 - i.e. show it belongs to P
 - Prove that no efficient algorithm exists
 - i.e. show it does not belong to P
- Complexity classes allow us to reason about sets of problems at a time, rather than each problem individually
 - If we can find more precise classes to organize problems into, we might be able to draw conclusions about the entire class
 - It may be easier to show a problem belongs to class C than to P , so it may help to show that $C \subseteq P$

Some problems in *EXP* seem “easier”

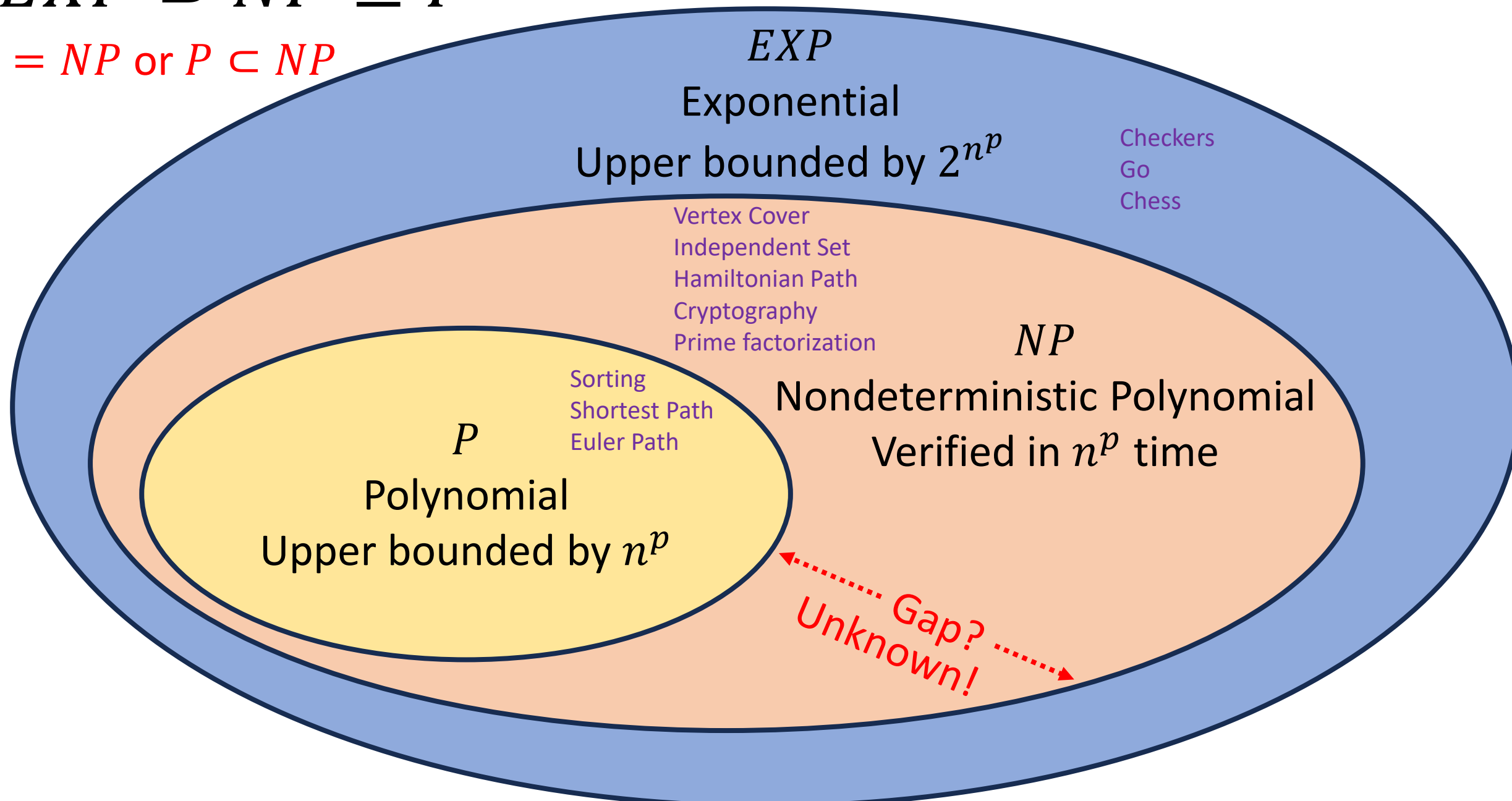
- There are some problems that we do not have polynomial time algorithms to solve, but provided answers are easy to check
- Hamiltonian Path:
 - It’s “hard” to look at a graph and determine whether it has a Hamiltonian Path
 - It’s “easy” to look at a graph and a candidate path together and determine whether THAT path is a Hamiltonian Path
 - It’s easy to **verify** whether a given path is a Hamiltonian path

Class NP

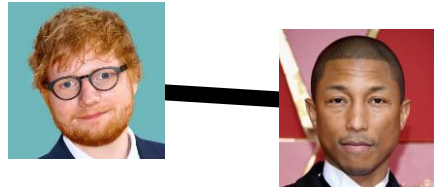
- NP
 - The set of problems for which a candidate solution can be verified in polynomial time
 - Stands for “Non-deterministic Polynomial”
 - Corresponds to algorithms that can guess a solution (if it exists), that solution is then verified to be correct in polynomial time
 - Can also think of as allowing a special operation that allows the algorithm to magically guess the right choice at each step of an exhaustive search
- $P \subseteq NP$
 - Why?

$$EXP \supset NP \supseteq P$$

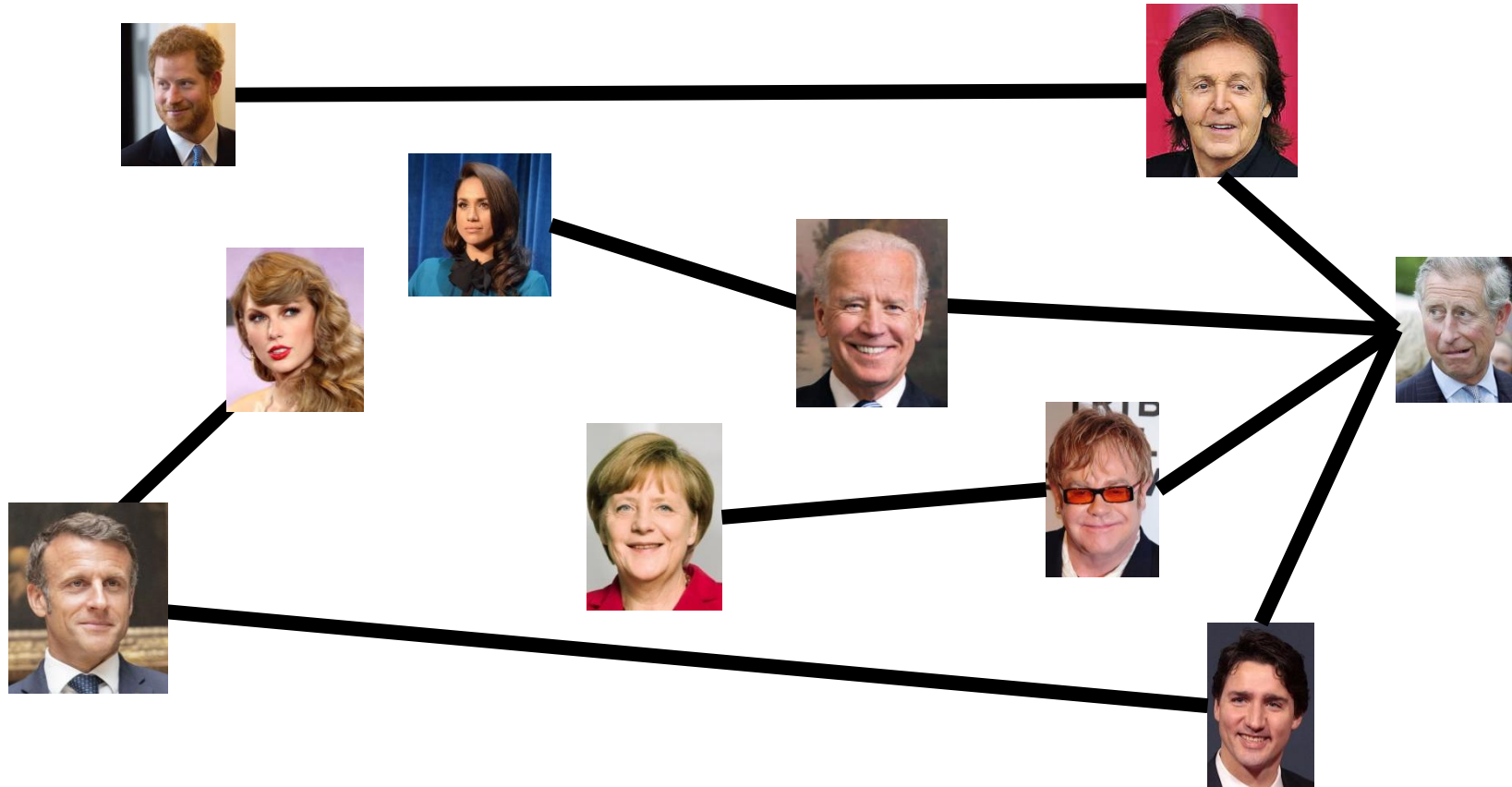
$P = NP$ or $P \subset NP$



Party Problem



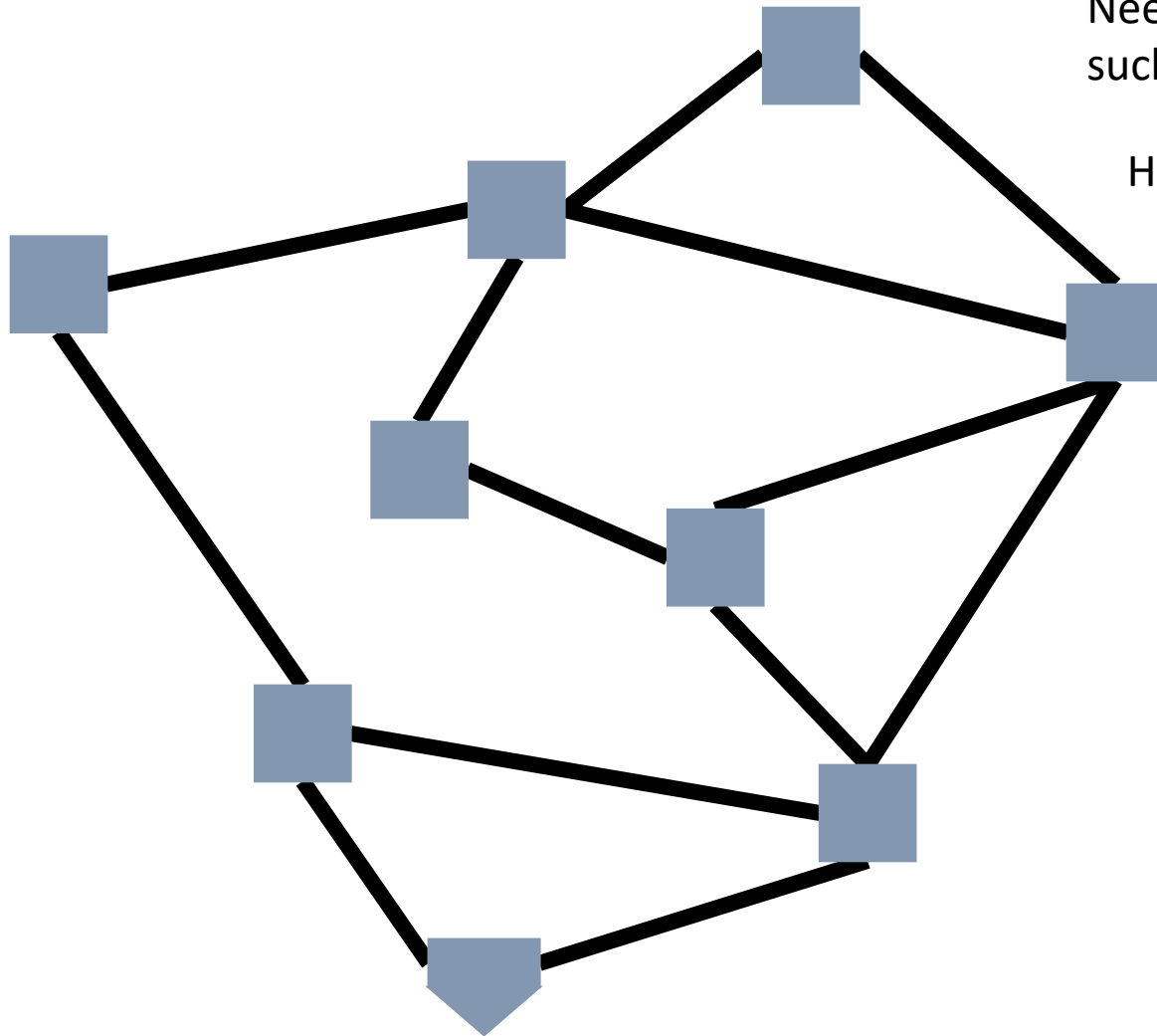
Draw Edges between people who don't get along
How many people can I invite to a party if everyone must get along?



Independent Set

- Independent set:
 - $S \subseteq V$ is an independent set if no two nodes in S share an edge
- Independent Set Problem:
 - Given a graph $G = (V, E)$ and a number k , determine whether there is an independent set S of size k

Generalized Baseball



Need to place defenders on bases such that every edge is defended

How many defenders would suffice?

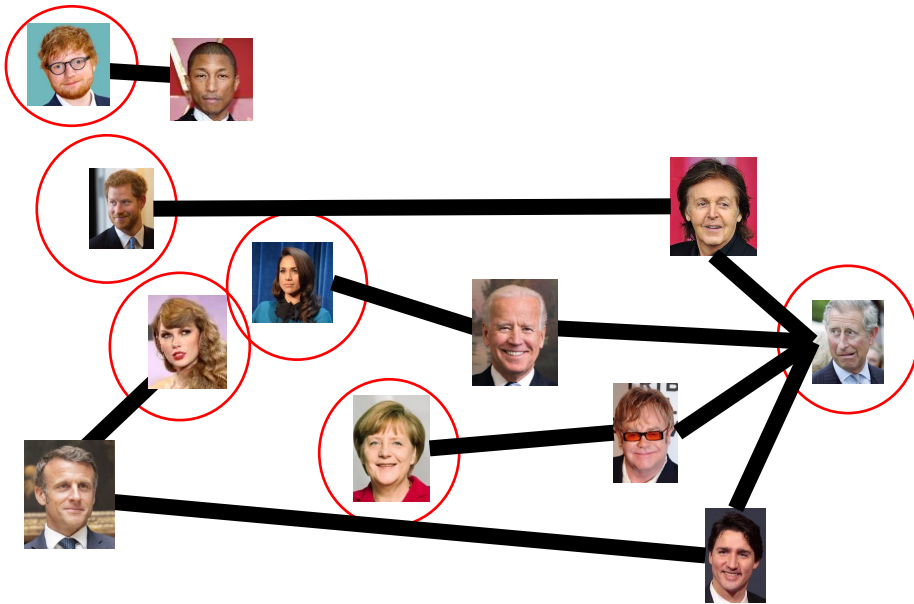
Vertex Cover

- Vertex Cover:
 - $C \subseteq V$ is a vertex cover if every edge in E has one of its endpoints in C
- Vertex Cover Problem:
 - Given a graph $G = (V, E)$ and a number k , determine if there is a vertex cover C of size k

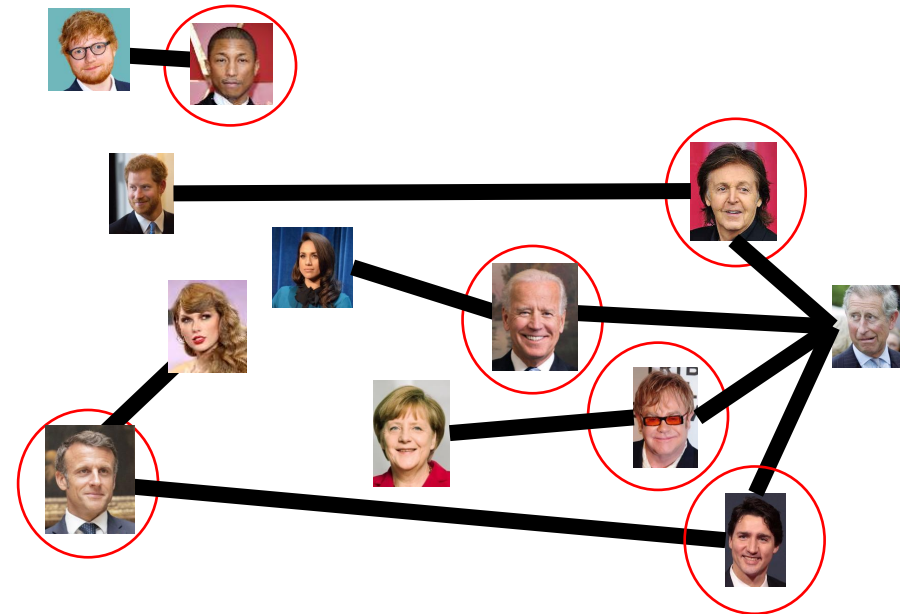
Transforming an IS into a VC

S is an independent set of G iff $V - S$ is a vertex cover of G

Independent Set



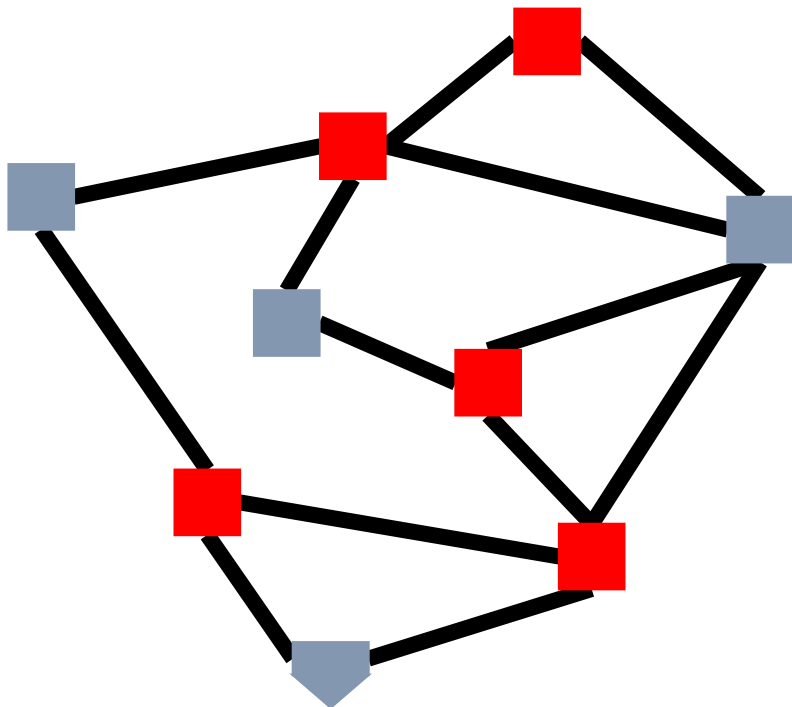
Vertex Cover



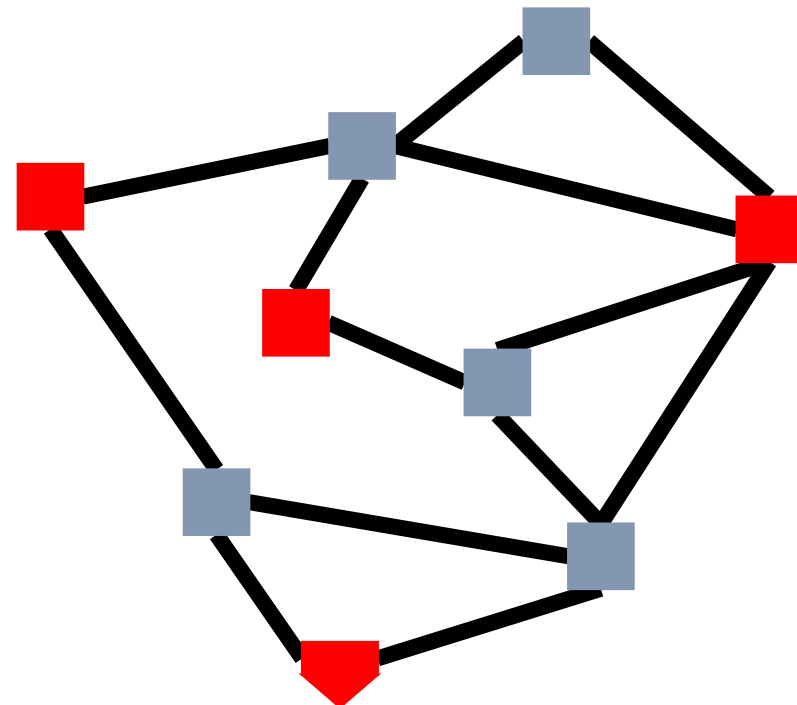
Transforming a VC into an IS

S is an independent set of G iff $V - S$ is a vertex cover of G

Vertex Cover



Independent Set



Solving Vertex Cover and Independent Set

- Algorithm to solve vertex cover
 - Input: $G = (V, E)$ and a number k
 - Output: True if G has a vertex cover of size k
 - Check if there is an Independent Set of G of size $|V| - k$
- Algorithm to solve independent set
 - Input: $G = (V, E)$ and a number k
 - Output: True if G has an independent set of size k
 - Check if there is a Vertex Cover of G of size $|V| - k$

Either both problems belong to P , or else neither does!

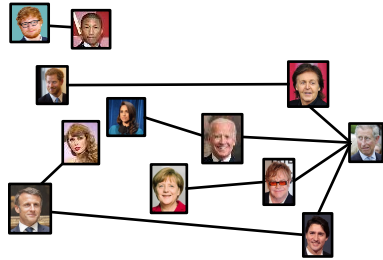
Reduction

- A strategy for creating algorithms
- Solve one problem by converting it into a different problem, then using an algorithm for that other problem.

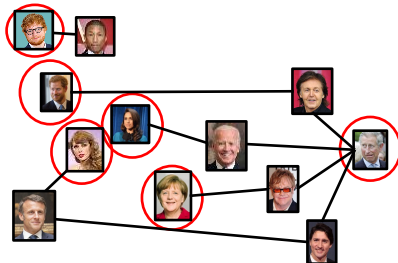
Independent Set Reduces To Vertex Cover

Independent Set Input

k



Independent Set Output



$O(V)$ Time

Relate Independent Set input to Vertex Cover output

Use same graph
Set $k = |V| - k$

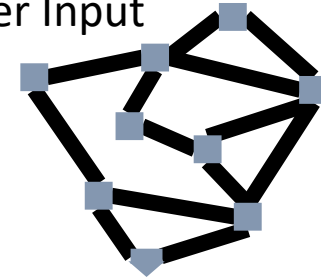
Relate Independent Set output to Vertex Cover output

Give same output

Reduction

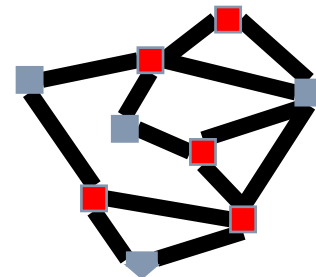
Vertex Cover Input

k



Any Algorithm for Vertex Cover

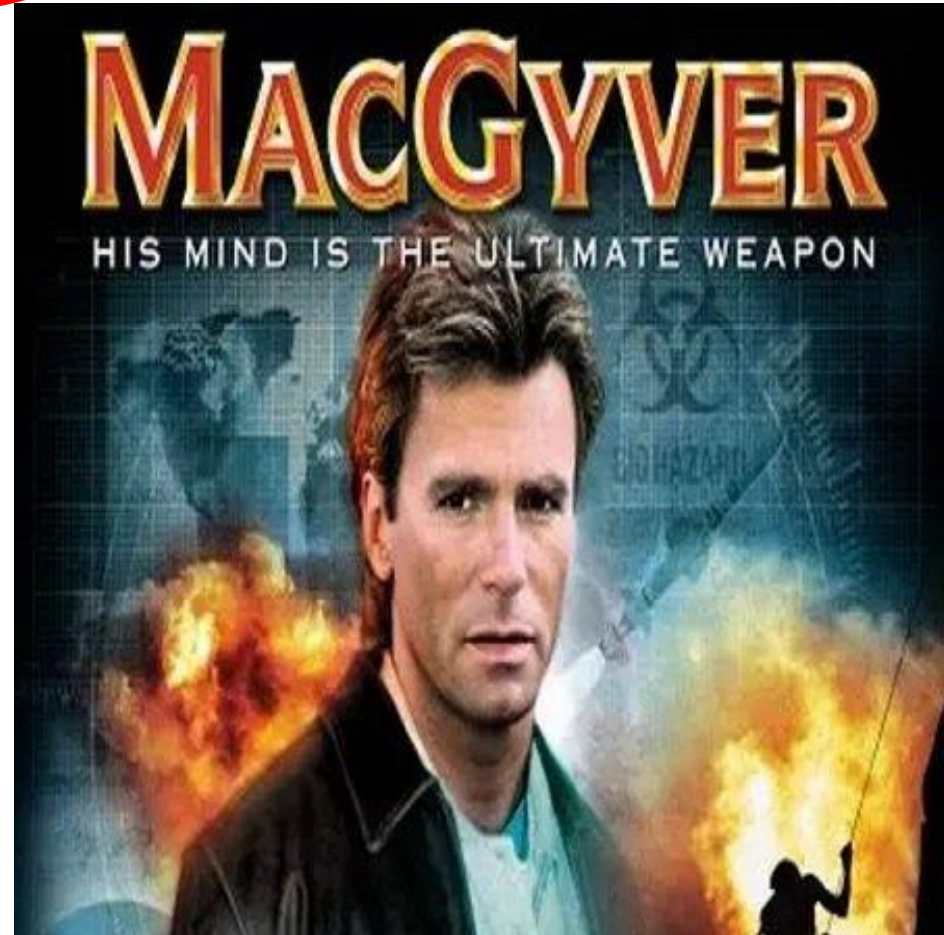
Vertex Cover Output



Reductions

Shows how two different problems relate to each other

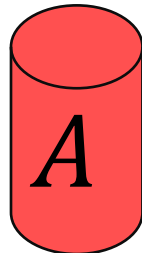
MOVIE TIME!



MacGyver's Reduction

Problem we don't know how to solve

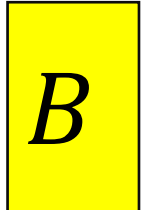
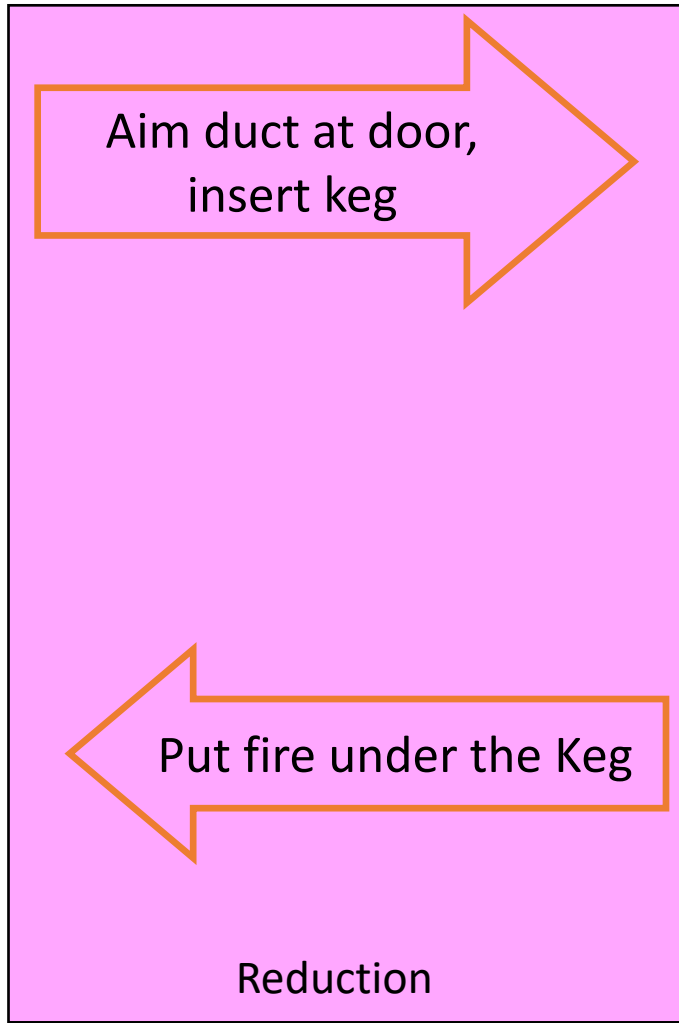
Problem we do know how to solve



Opening a door



Solution for *A*
Keg cannon
battering ram



Lighting a fire



Solution for *B*
Alcohol, wood,
matches

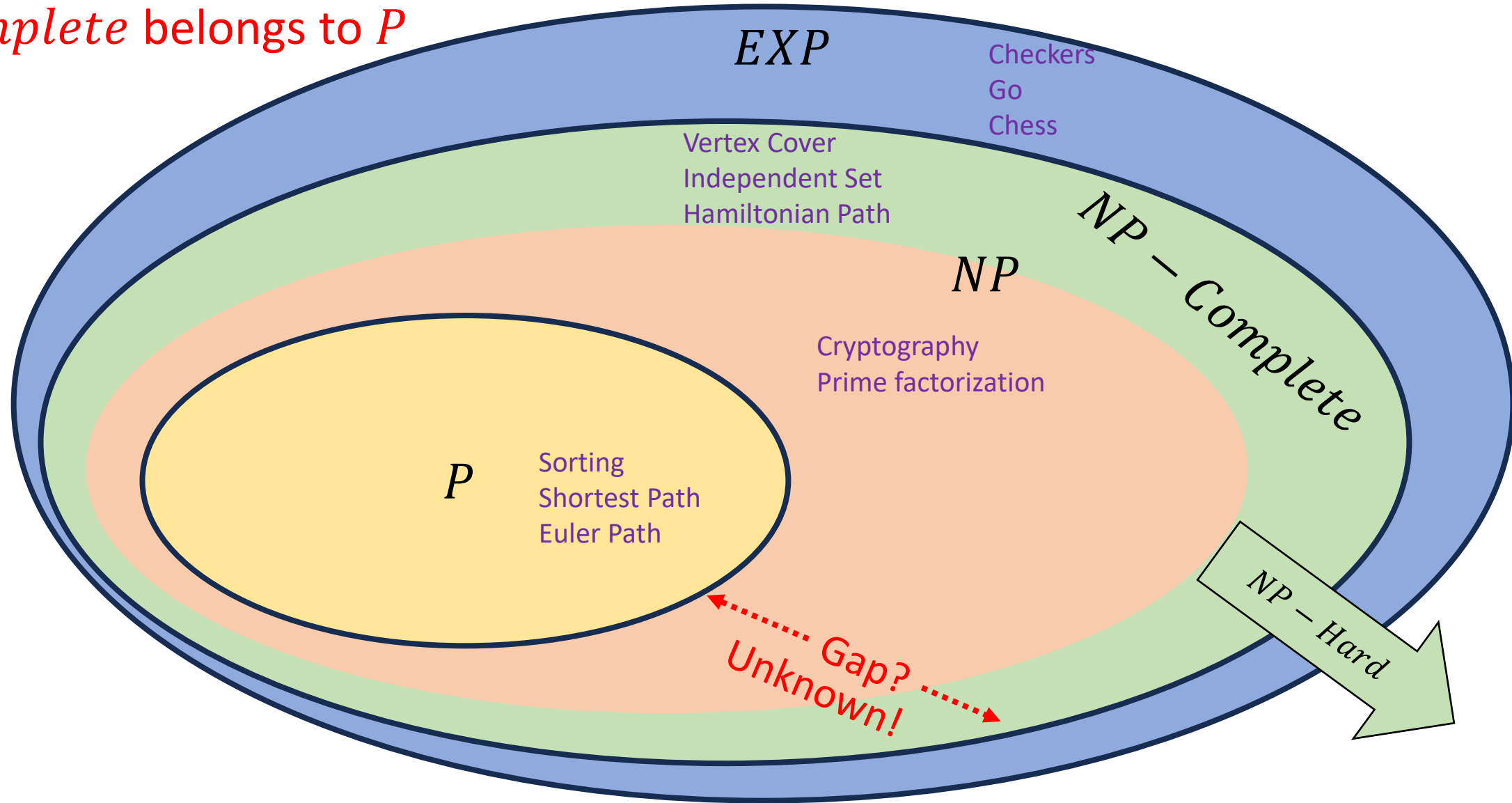


NP-Complete

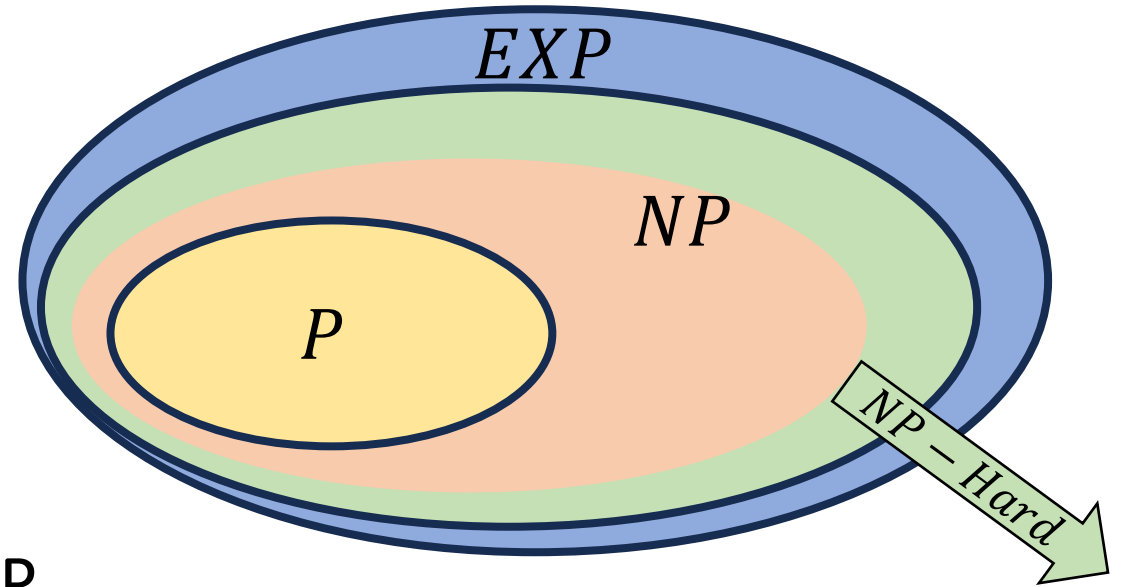
- A set of “together they stand, together they fall” problems
- The problems in this set either all belong to P , or none of them do
- Intuitively, the “hardest” problems in NP
- Collection of problems from NP that can all be “transformed” into each other in polynomial time
 - Like we could transform independent set to vertex cover, and vice-versa
 - We can also transform vertex cover into Hamiltonian path, and Hamiltonian path into independent set, and ...

$$EXP \supset NP - Complete \supseteq NP \supseteq P$$

$P = NP$ iff some problem from
 $NP - Complete$ belongs to P



NP-Hard

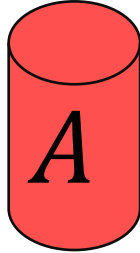


- How can we try to figure out if $P=NP$?
- Identify problems at least as “hard” as NP
 - If any of these “hard” problems can be solved in polynomial time, then all NP problems can be solved in polynomial time.
- Definition: $NP-Hard$:
 - B is $NP-Hard$ provided EVERY problem within NP reduces to B in polynomial time

For every NP problem

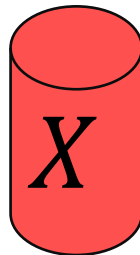
NP-Hard Idea

Any NP Problem

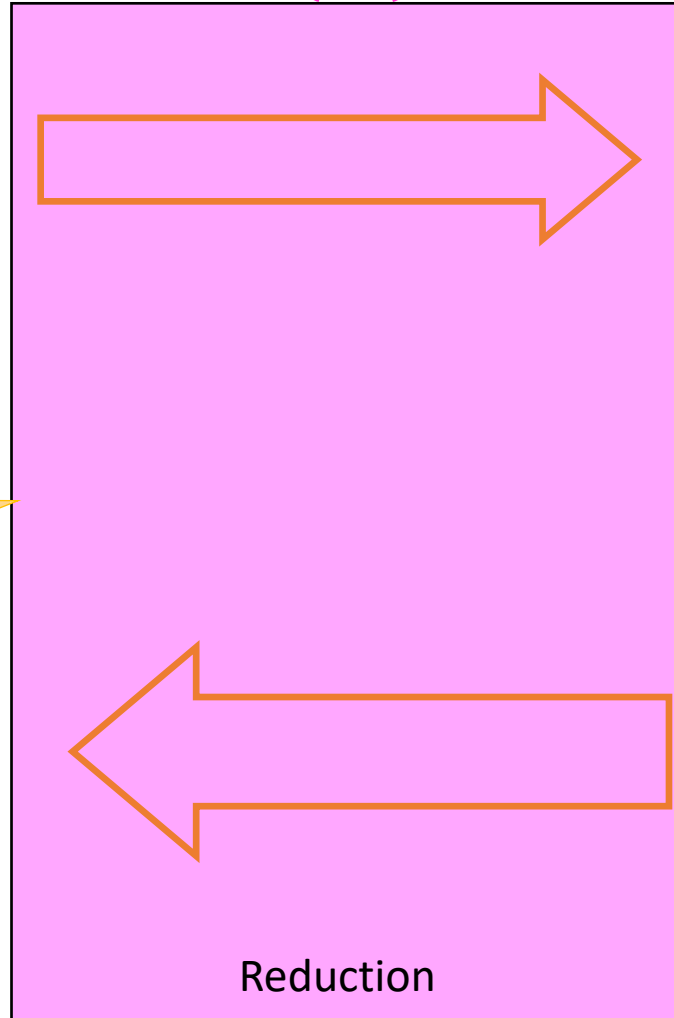


There exists a polynomial-time reduction to each NP-Hard Problem

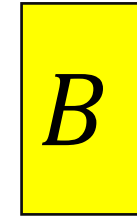
Solution for *A*



$O(n^p)$



An NP-Hard Problem



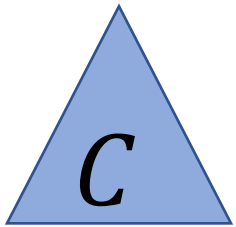
Solution for *B*



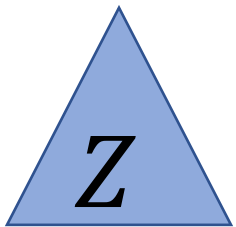
So if this was $O(n^p)$ we can solve any NP problem in polynomial time

Showing NP-Hardness

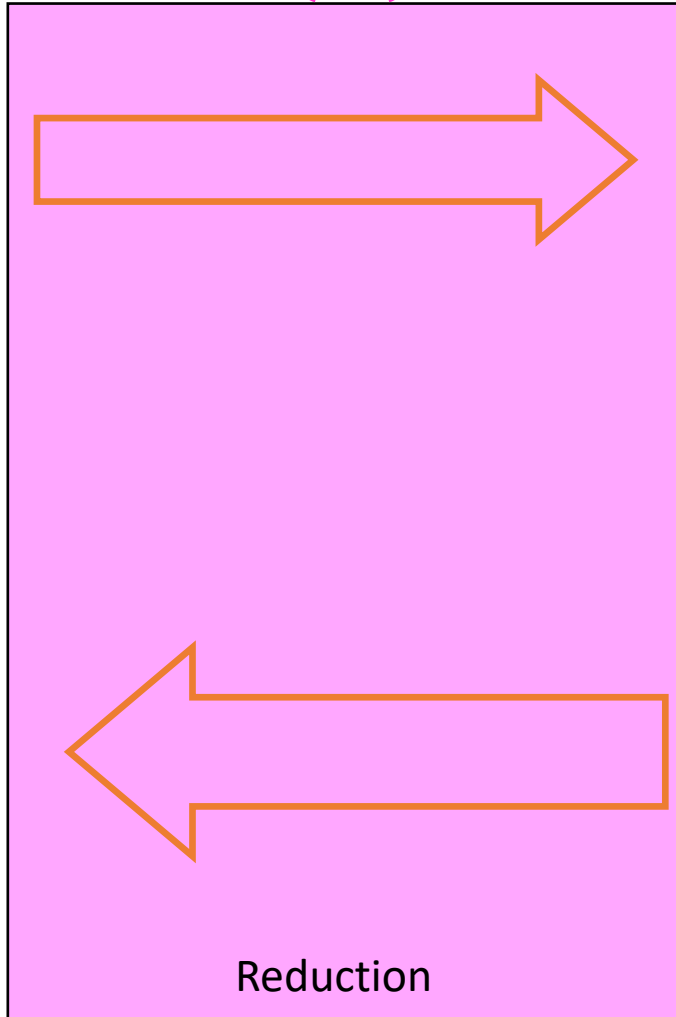
Any NP Problem



Solution for C



$O(n^p)$



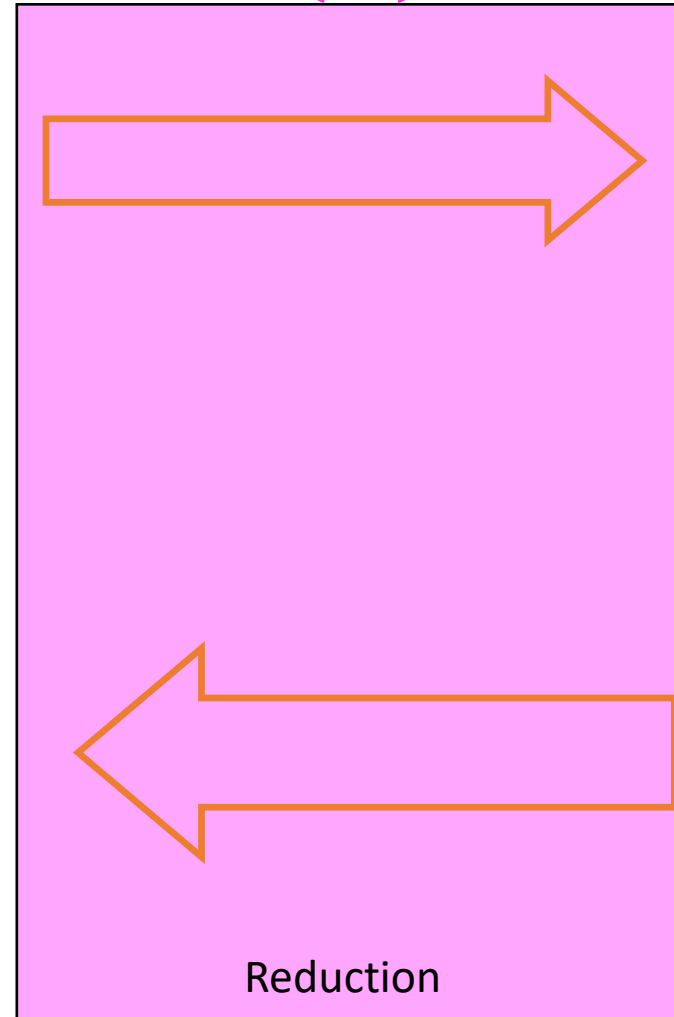
A First NP-Hard Problem



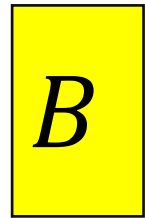
Solution for A



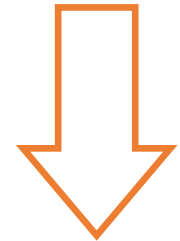
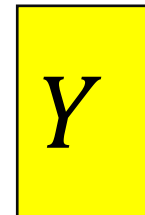
$O(n^p)$



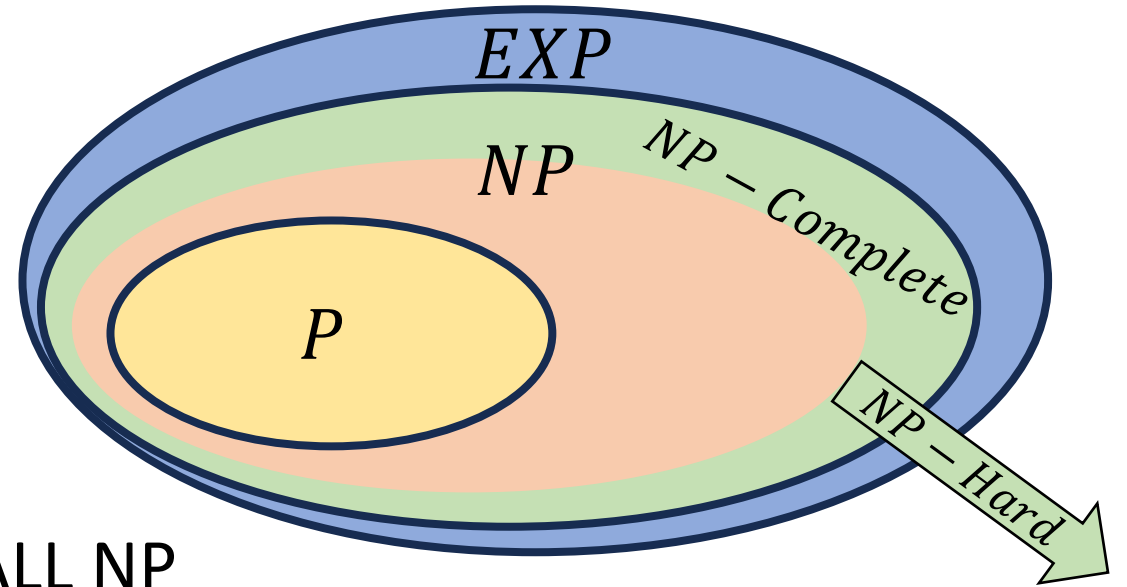
A new NP-Hard Problem



Solution for B



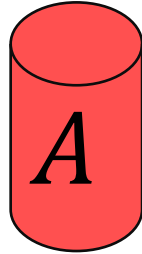
NP-Complete



- “Together they stand, together they fall”
- Problems solvable in polynomial time iff ALL NP problems are
- NP-Complete = $NP \cap NP\text{-Hard}$
- **How to show a problem is NP-Complete?**
 - Show it belongs to NP
 - Give a polynomial time verifier
 - Show it is NP-Hard
 - Give a reduction from another NP-H problem

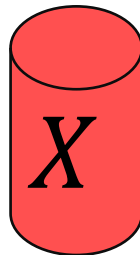
NP-Completeness

Any NP-Complete Problem

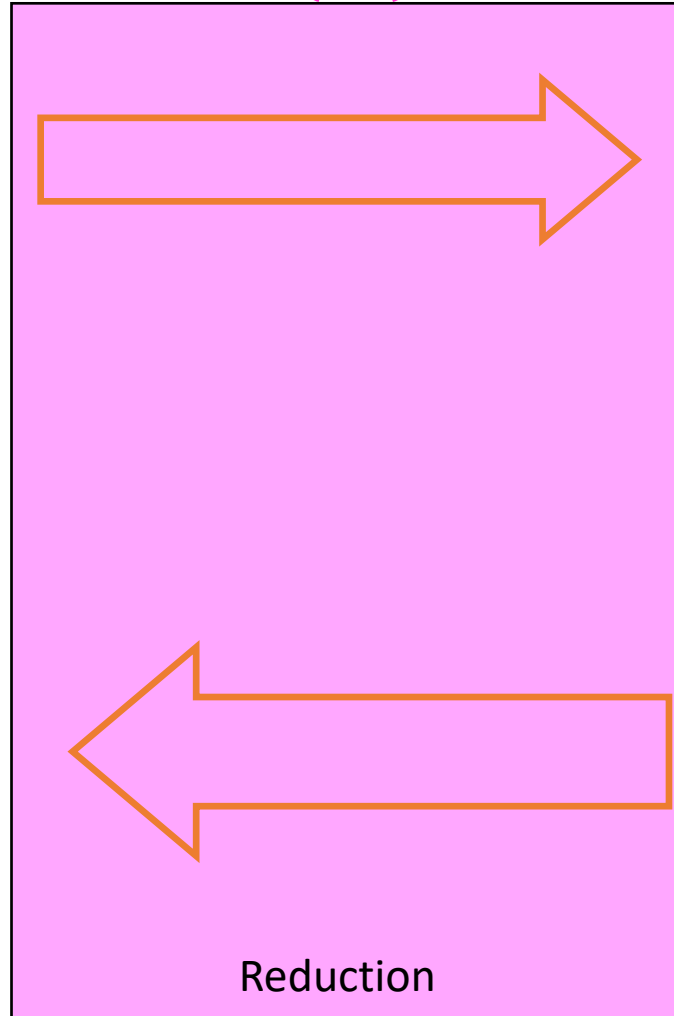


Then this could be done in polynomial time

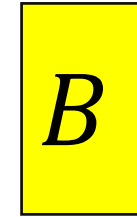
Solution for A



$O(n^p)$

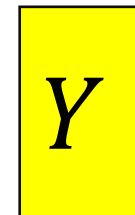


Any other NP-Complete Problem



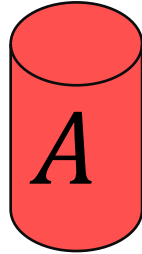
If this could be done in polynomial time

Solution for B



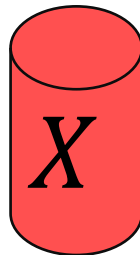
NP-Completeness

Any NP-Complete Problem

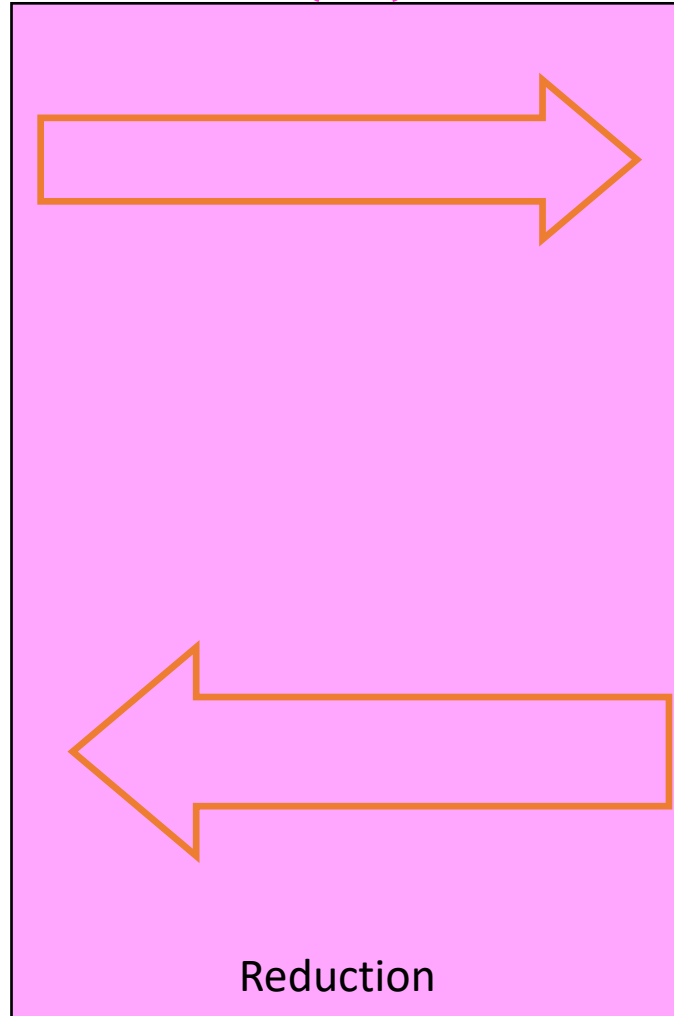


If this cannot be done in polynomial time

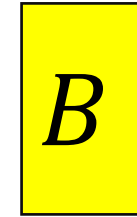
Solution for A



$O(n^p)$

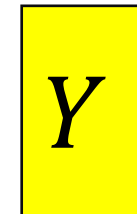


Any other NP-Complete Problem



Then this cannot be done in polynomial time

Solution for B



Overview

- Problems not belonging to P are considered intractable
- The problems within NP have some properties that make them seem like they might be tractable, but we've been unsuccessful with finding polynomial time algorithms for many
- The class $NP - Complete$ contains problems with the properties:
 - All members are also members of NP
 - All members of NP can be transformed into every member of $NP - Complete$
 - Because they are both NP and $NP - Hard$
 - If any one member of $NP - Complete$ belongs to P , then $P = NP$
 - If any one member of $NP - Complete$ is outside of P , then $P \neq NP$

Why should YOU care?

- If you can find a polynomial time algorithm for any *NP – Complete* problem then:
 - You will win \$1million
 - You will win a Turing Award
 - You will be world famous
 - You will have done something that no one else on Earth has been able to do in spite of the above!
- If you are told to write an algorithm a problem that is *NP – Complete*
 - You can tell that person everything above to set expectations
 - Change the requirements!
 - **Approximate the solution:** Instead of finding a path that visits every node, find a path that visits at least 75% of the nodes
 - **Add Assumptions:** problem might be tractable if we can assume the graph is acyclic, a tree
 - **Use Heuristics:** Write an algorithm that’s “good enough” for small inputs, ignore edge cases

Why should YOU care?

- The entire field of cryptography relies on it (nearly at least)
 - Requires decrypting with a key is easier than decrypting without a key
 - This is strongly related to requiring a difference in difficulty between verifying a candidate solution and finding a solution in the first place
- If $P \neq NP$
 - Some problems remain intractable
 - Cryptography persists
- If $P = NP$
 - We may get efficient solutions for important problems
 - Cryptography is potentially doomed.

Does $P=NP$?

	$P \neq NP$	$P=NP$	Ind	DC	DK	DK and DC	other
2002	61 (61%)	9 (9%)	4 (4%)	1 (1%)	22 (22%)	0 (0%)	3 (3%)
2012	126 (83%)	12 (9%)	5 (3%)	5 (3%)	1 (0.66%)	1 (0.66%)	1 (0.66%)
2019	109 (88%)	15 (12%)	0	0	0	0	0

When Will P=NP be resolved?

	02-09	10-19	20-29	30-39	40-49	50-59	60-69	70-79
2002	5 (5%)	12 (12%)	13 (13%)	10 (10%)	5 (5%)	12 (12%)	4 (4%)	0 (0%)
2012	0 (0%)	2 (1%)	17 (11%)	18 (12%)	5 (3%)	10 (6.5%)	10 (6.5%)	9 (6%)
2019	0 (0%)	0 (0%)	26 (22%)	20 (17%)	14 (12%)	9 (7%)	7 (6%)	5 (4%)

	80-89	90-99	100-109	110-119	150-159	2200-3000	4000-4100
2002	1 (1%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	5 (5%)	0 (0%)
2012	4 (3%)	5 (3%)	2 (1.2%)	5 (3%)	2 (1.2%)	3 (2%)	3 (2%)
2019	0 (0%)	0 (0%)	1 (0.8%)	10 (12%)	10 (12%)	1 (0.8%)	11 (9%)

	Long Time	Never	Don't Know	Sooner than 2100	Later than 2100
2002	0 (0%)	5 (5%)	21 (21%)	62 (62%)	17 (17%)
2012	22 (14%)	5 (3%)	8 (5%)	81 (53%)	63 (41%)
2019	7 (6%)	11 (9%)	0 (0%)	84 (66%)	40 (34%)

Notable Statements on P vs NP

Scott Aaronson I believe $P \neq NP$ on basically the same grounds that I think I won't be devoured tomorrow by a 500-foot-tall robotic marmoset from Venus, despite my lack of proof in both cases.

Suggested rephrased question:

will humans manage to prove $P \neq NP$ before they either kill themselves out or are transcended by superintelligent cyborgs? And if the latter, will the cyborgs be able to prove $P \neq NP$?

Neil Immerman $P \neq NP$ will be resolved somewhere between 2017 and 2034, using some combination of logic, algebra, and combinatorics.

Donald Knuth: (Retired from Stanford) It will be solved by either 2048 or 4096. I am currently somewhat pessimistic. The outcome will be the truly worst case scenario: namely that someone will prove " $P=NP$ because there are only finitely many obstructions to the opposite hypothesis"; hence there will exist a polynomial time solution to SAT but we will never know its complexity!