Unit 7: Theory

Dave Abel

March 16th, 2016

As Promised:

Any Midterm questions?

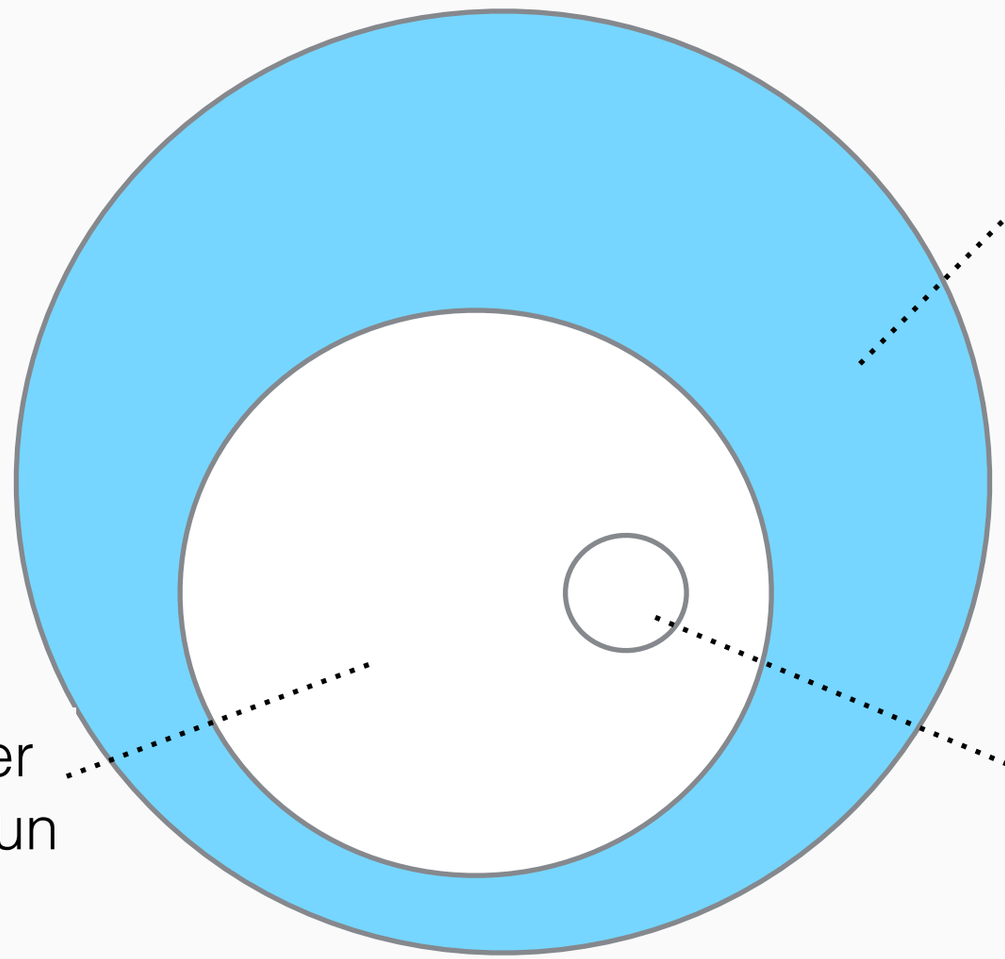
Growth Rates

- So far, we've talked about the growth rates of *algorithms*, e.g. Binary Search, Selection Sort:
  - Linear Search: N
  - Binary Search: $log(N)$
  - Selection Sort: N^2
  - Build the Truth Table: 2^N

Growth Rates

- So far, we've talked about the growth rates of *algorithms*, e.g. Binary Search, Selection Sort:
  - Linear Search: *N*
  - Binary Search: *log(N)*
  - Selection Sort: N^2
  - Build the Truth Table: 2^N
- **But don't we care about *problems*?**

Growth Rates

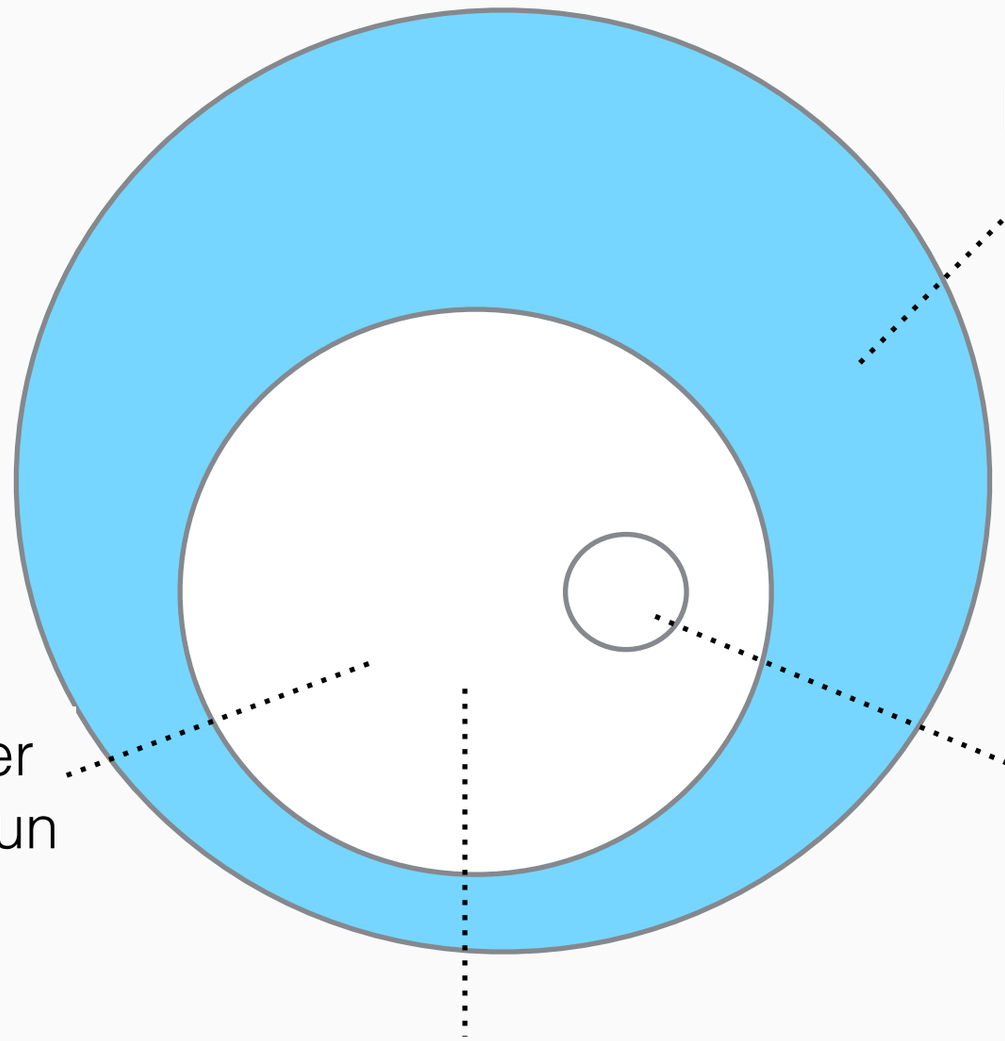


Things that can be computed, period.

Things a regular computer can compute before the sun goes supernova

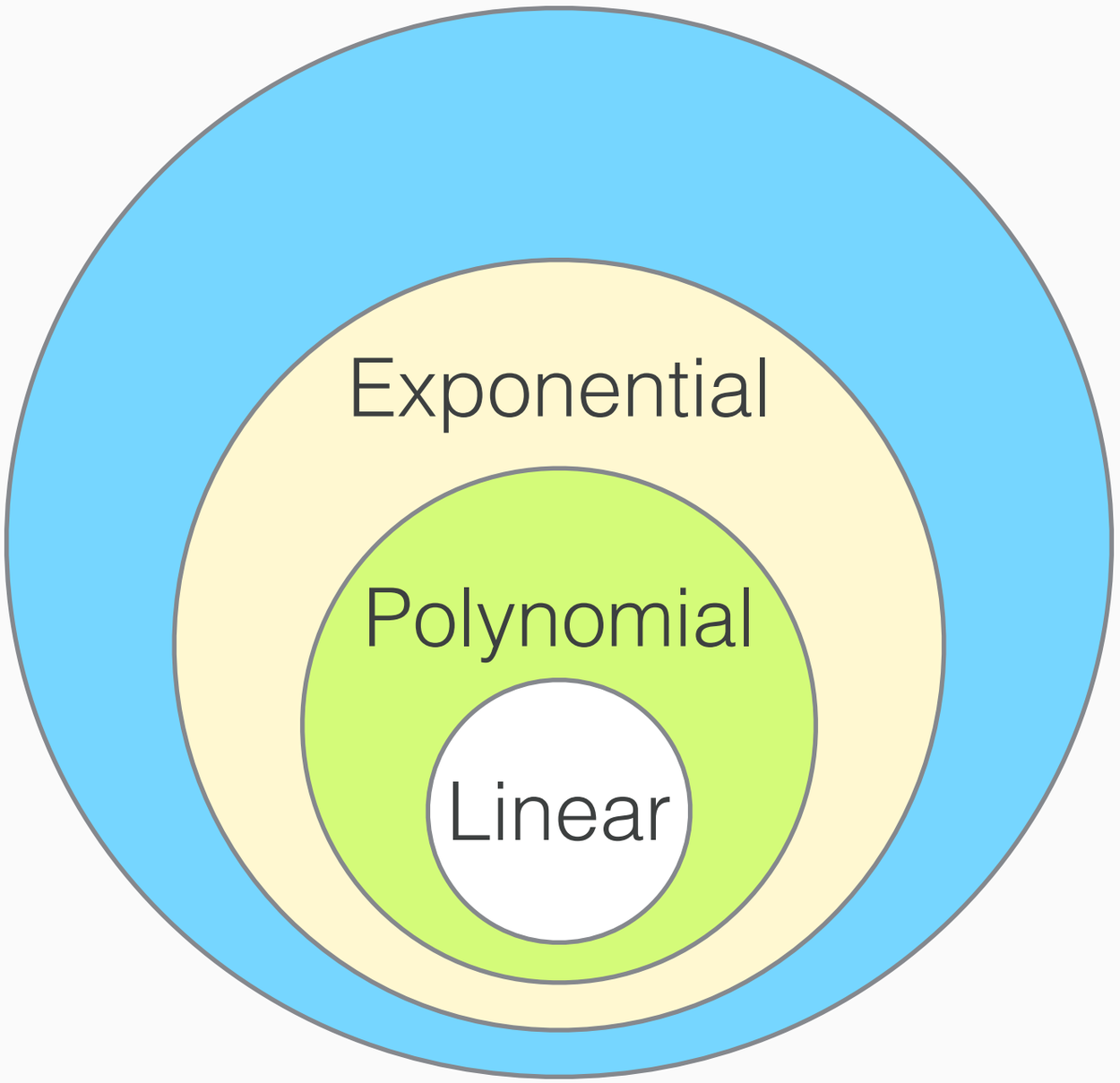
Things a domino computer could compute before the sun goes supernova

Growth Rates

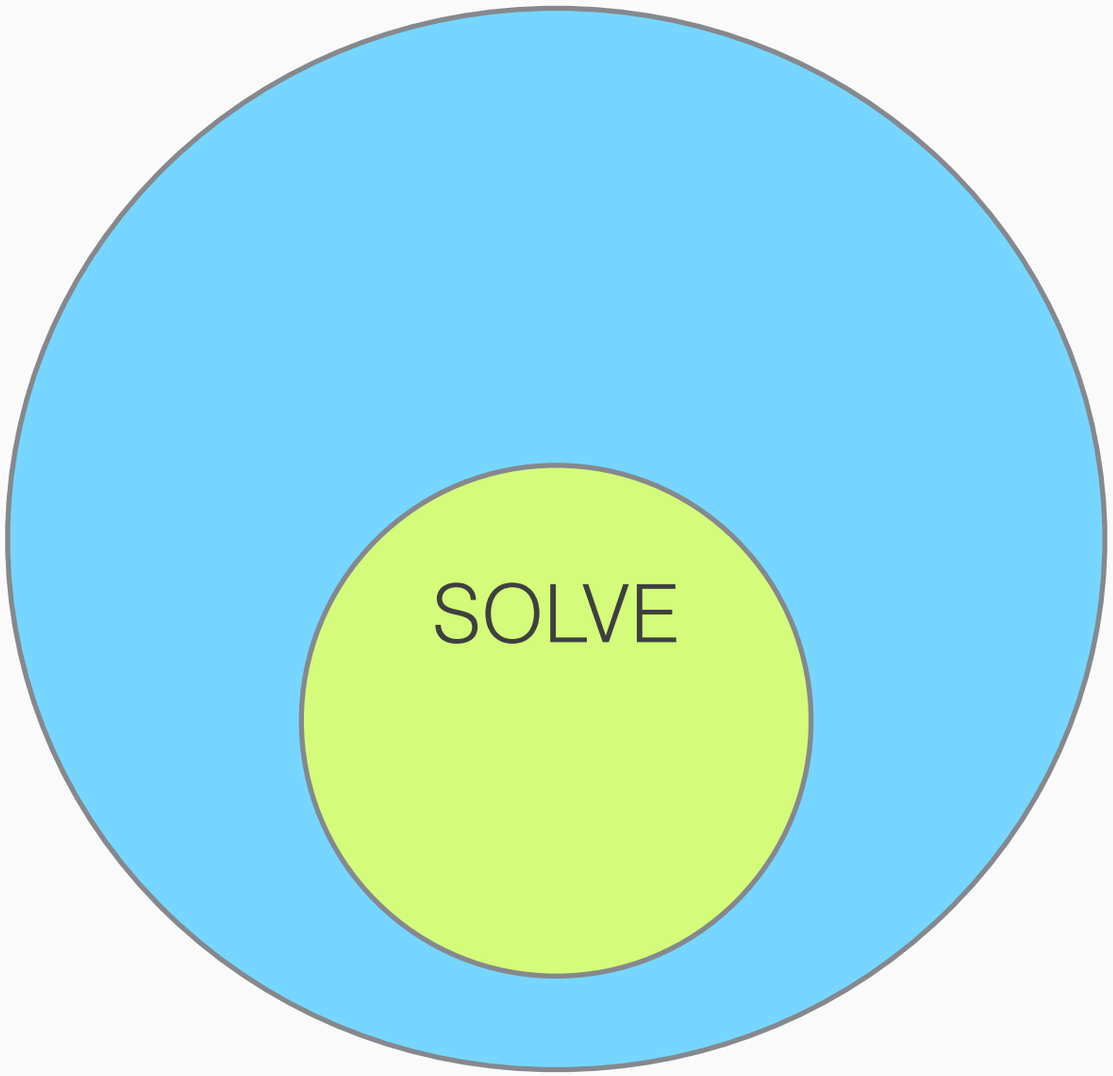


Q: What problems can we solve in a reasonable amount of time?

Growth Rates



SOLVE



Q: Can we solve a problem efficiently?

A: Is it in SOLVE?

Reminder:

- Problem Specification:
  - INPUT: some things
  - OUTPUT: some true stuff about the things

Reminder:

- Problem Specification:
  - INPUT: some things
  - OUTPUT: some true stuff about the things
- Example:
  - INPUT: A Sudoku board
  - OUTPUT: Solution to the Sudoku board

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

Another View: Verification

- Verification Example:
  - INPUT: An empty Sudoku board, a proposed solution to that Sudoku board
  - OUTPUT: True if the Sudoku board is a correct solution

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

Another View: Verification

- Another Verification Example:
  - INPUT: An empty Crossword, a proposed solution to that Crossword
  - OUTPUT: True if the filled out Crossword board is a correct solution

Another View: Verification

- Another Verification Example:
  - INPUT: A list, a proposed sorting of that list
  - OUTPUT: True if the proposed sorting is actually in sorted order.

Alpha Go

- In light of recent events, consider how *making the perfect single move* in the Game Go can be pitched as a verification problem!

INPUT: A configuration of the Go board, a Go move

OUTPUT: True if the move is the perfect move.

Another Class: VERIFY

Definition: The class of problems VERIFY is the set of problems where we can **efficiently** *verify* solutions

Another Class: VERIFY

Definition: The class of problems VERIFY is the set of problems where we can **efficiently** *verify* solutions

Q: Can we efficiently verify Sudoku?

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

Another Class: VERIFY

Definition: The class of problems VERIFY is the set of problems where we can **efficiently** *verify* solutions

Q: Can we efficiently verify Sudoku?

A: Totally! Just make sure each cell, row, column has 1-9.

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

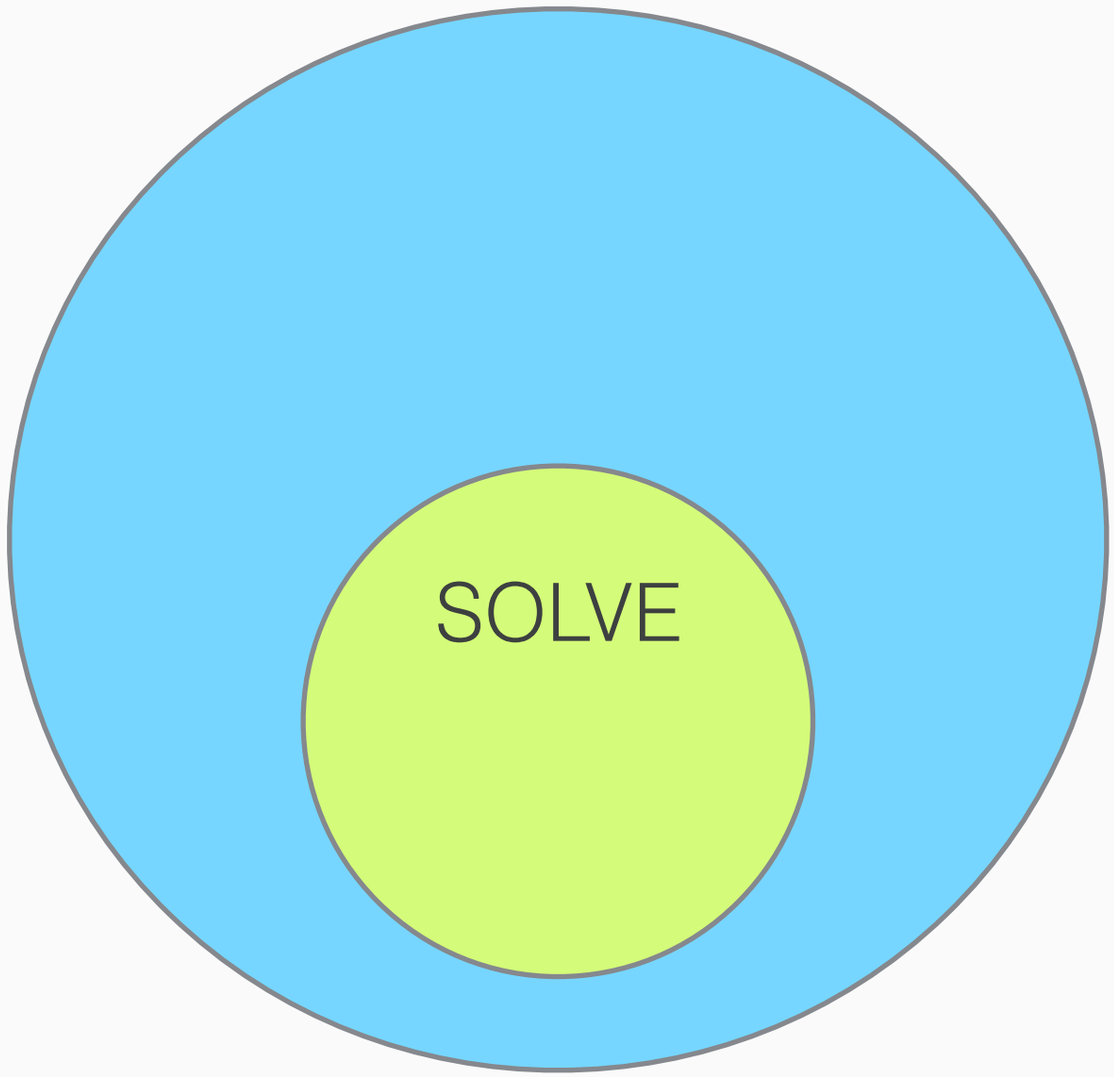
Another Class: VERIFY

Definition: The class of problems VERIFY is the set of problems where we can **efficiently** *verify* solutions

Q: Can we efficiently verify Go?

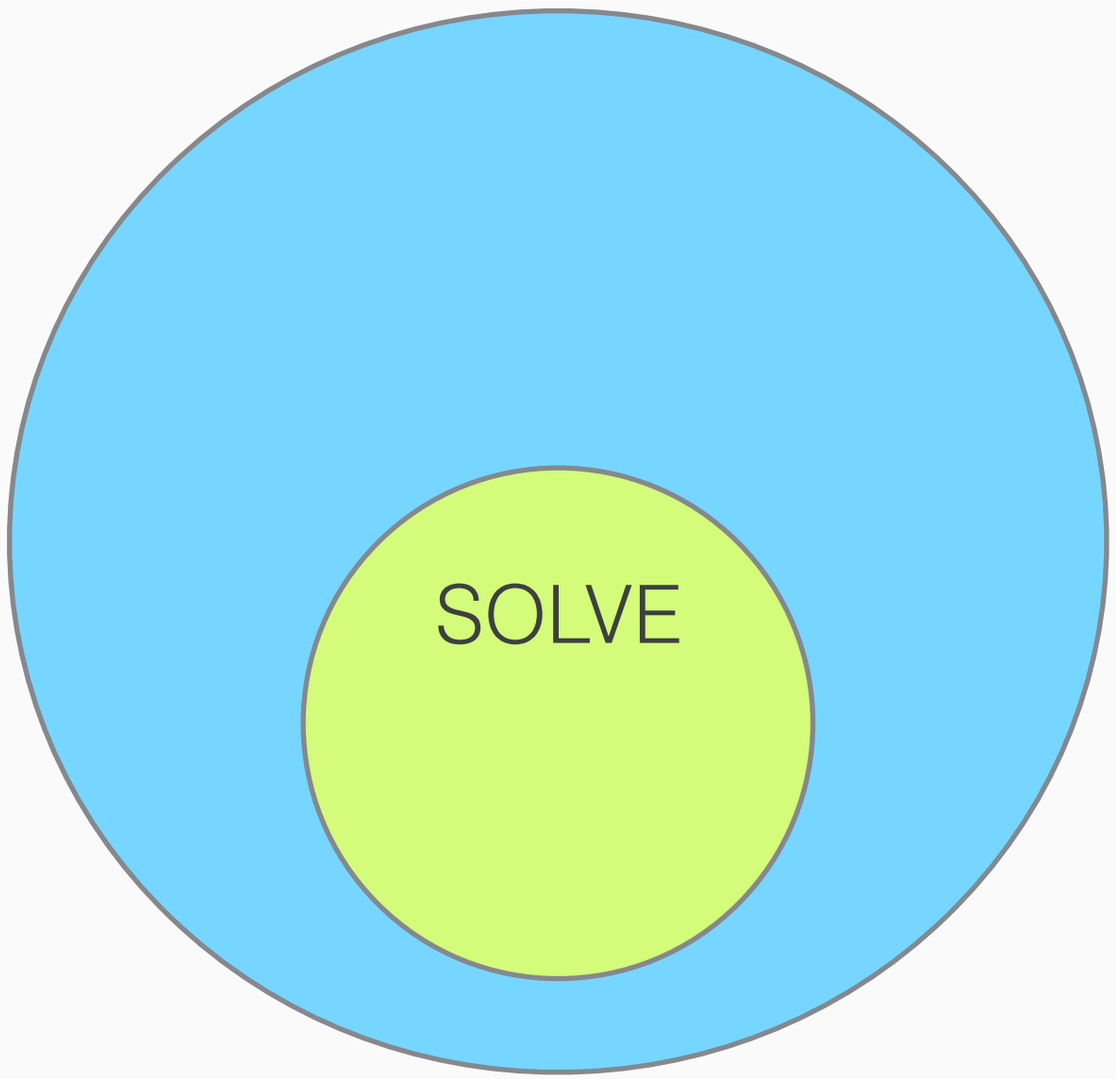
A: Definitely not!

Some Clicker Questions!



Q: Which of these describes the SOLVE class?

Some Clicker Questions!



[A] Problems that we think are efficiently solvable

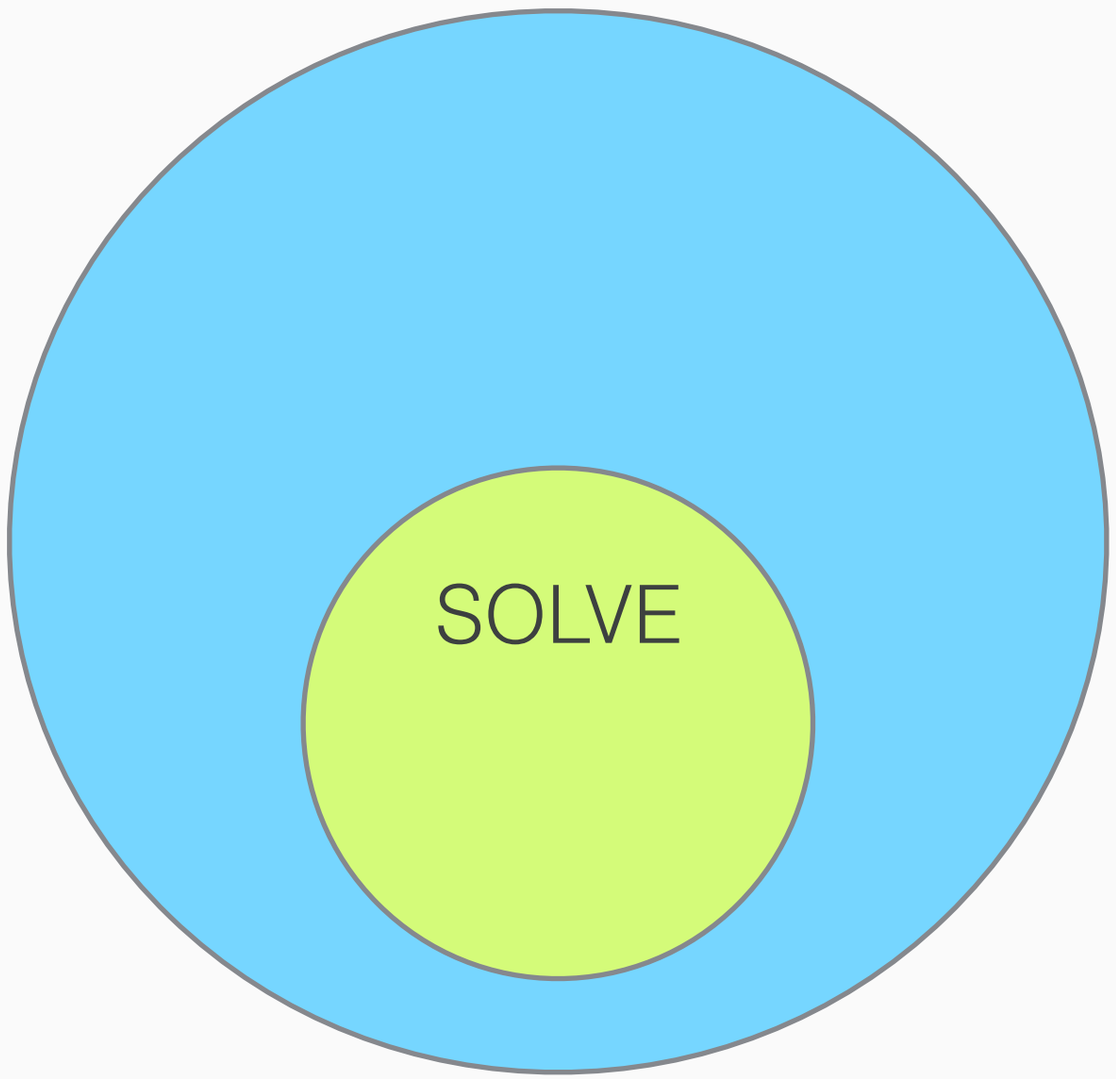
[B] Algorithms whose growth rate is *N* or faster

[C] Problems where candidate solutions can be verified easily

[D] I'm confused

Q: Which of these describes the SOLVE class?

Some Clicker Questions!



[A] Problems that we think are efficiently SOLVable

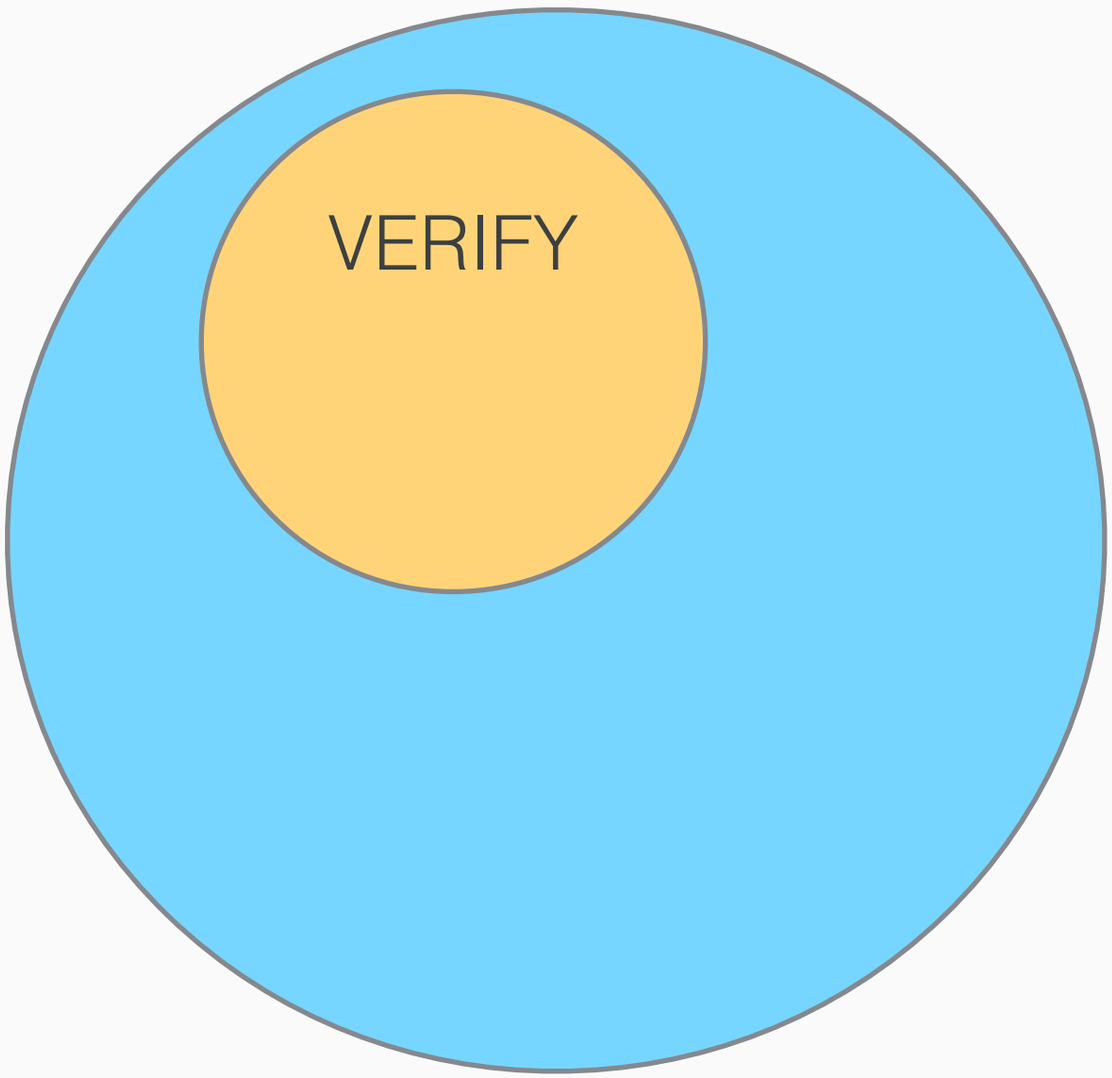
[B] Algorithms whose growth rate is *N* or faster

[C] Problems where candidate solutions can be verified easily

[D] I'm confused

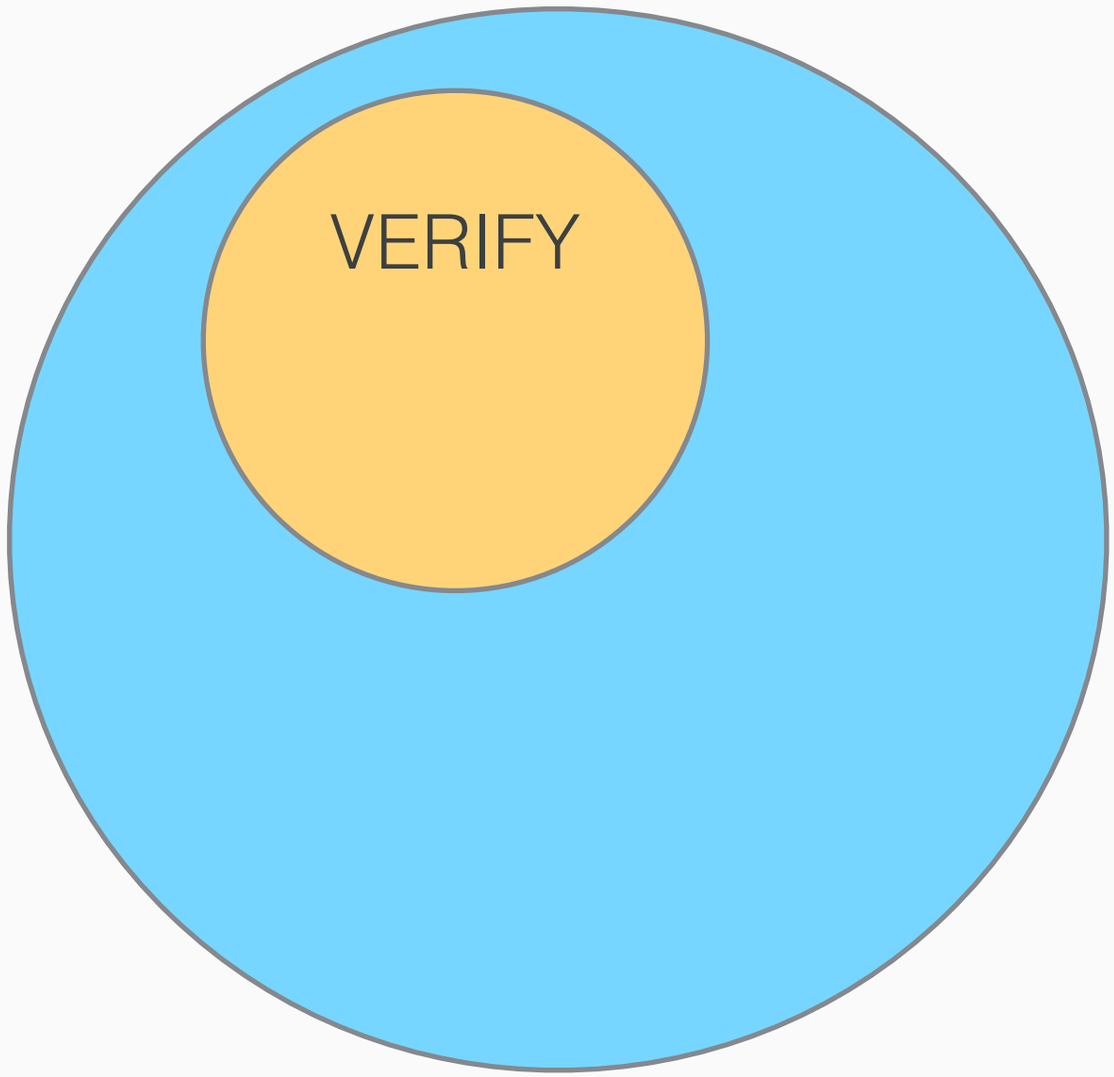
Q: Which of these describes the SOLVE class?

Some Clicker Questions!



Q: Which of these describes the VERIFY class?

Some Clicker Questions!



[A] Problems that we think are efficiently solvable

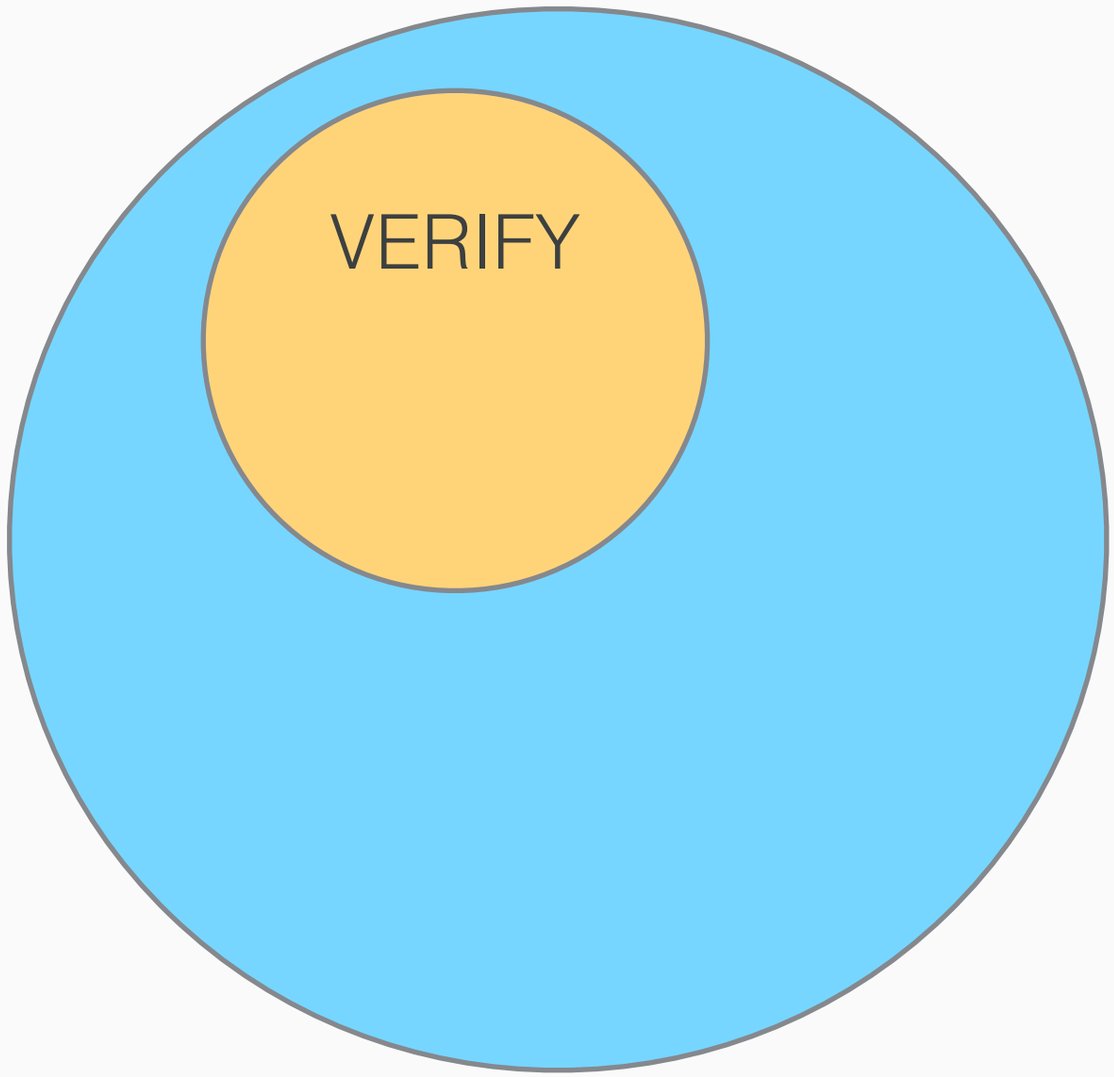
[B] Algorithms whose growth rate is *N* or faster

[C] Problems where candidate solutions can be verified easily

[D] I'm confused

Q: Which of these describes the VERIFY class?

Some Clicker Questions!



[A] Problems that we think are efficiently solvable

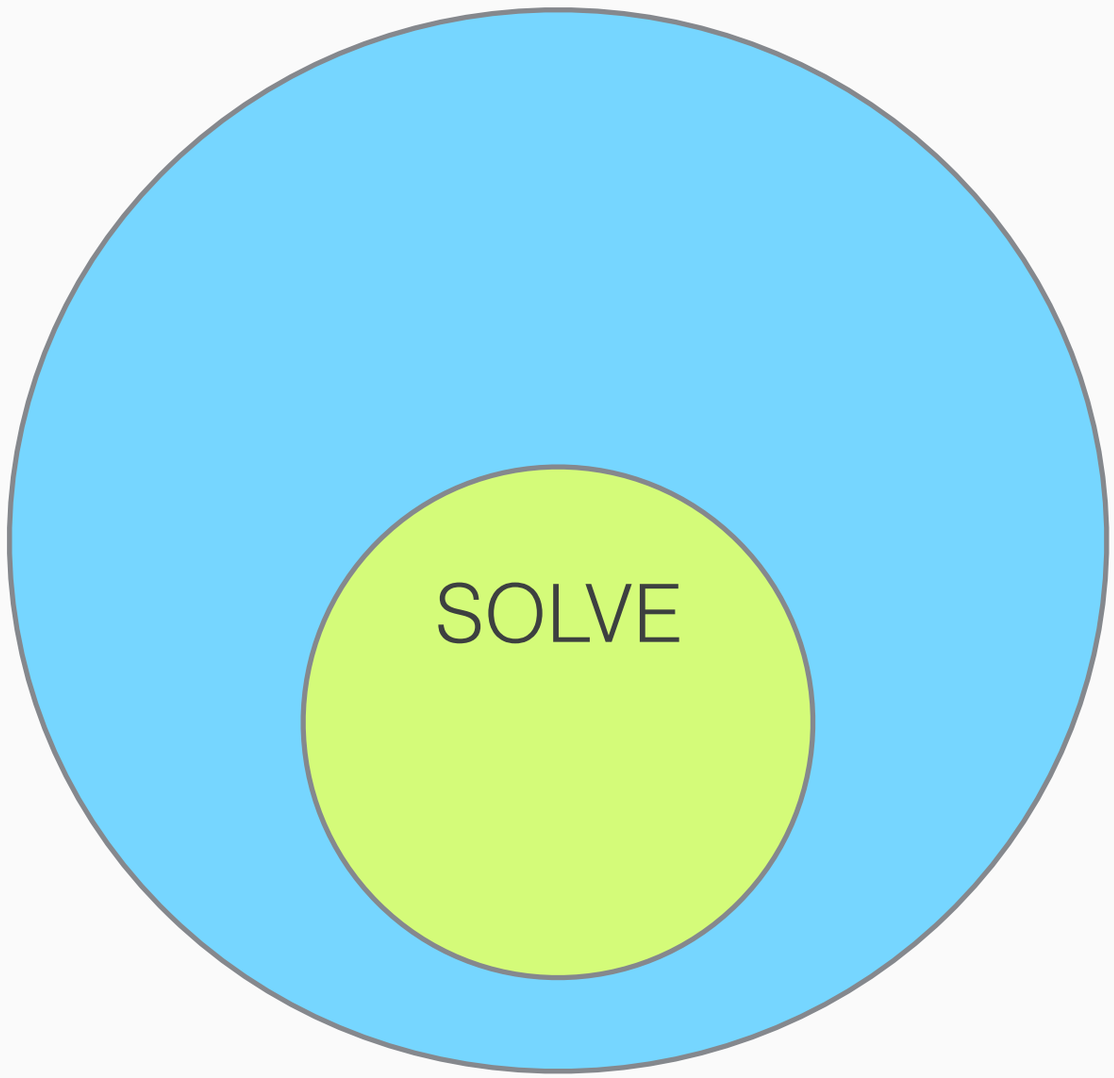
[B] Algorithms whose growth rate is *N* or faster

[C] Problems where candidate solutions can be VERIFied easily

[D] I'm confused

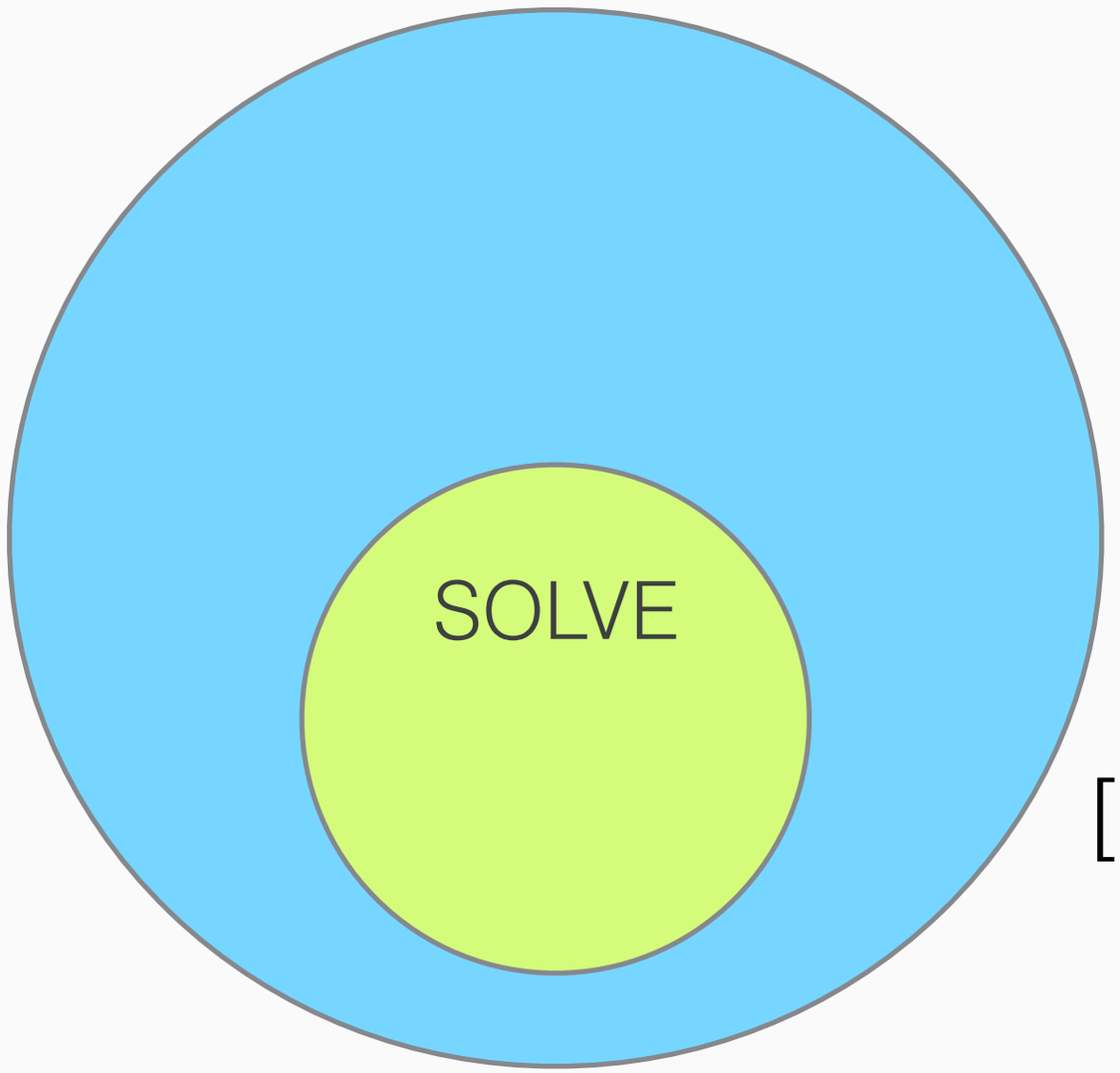
Q: Which of these describes the VERIFY class?

Clicker Question!



Q: Given *any* problem known to be in SOLVE, do we know anything about its status relative to VERIFY*?*

Clicker Question!



Q: Given *any* problem known to be in SOLVE, do we know anything about its status relative to VERIFY*?*

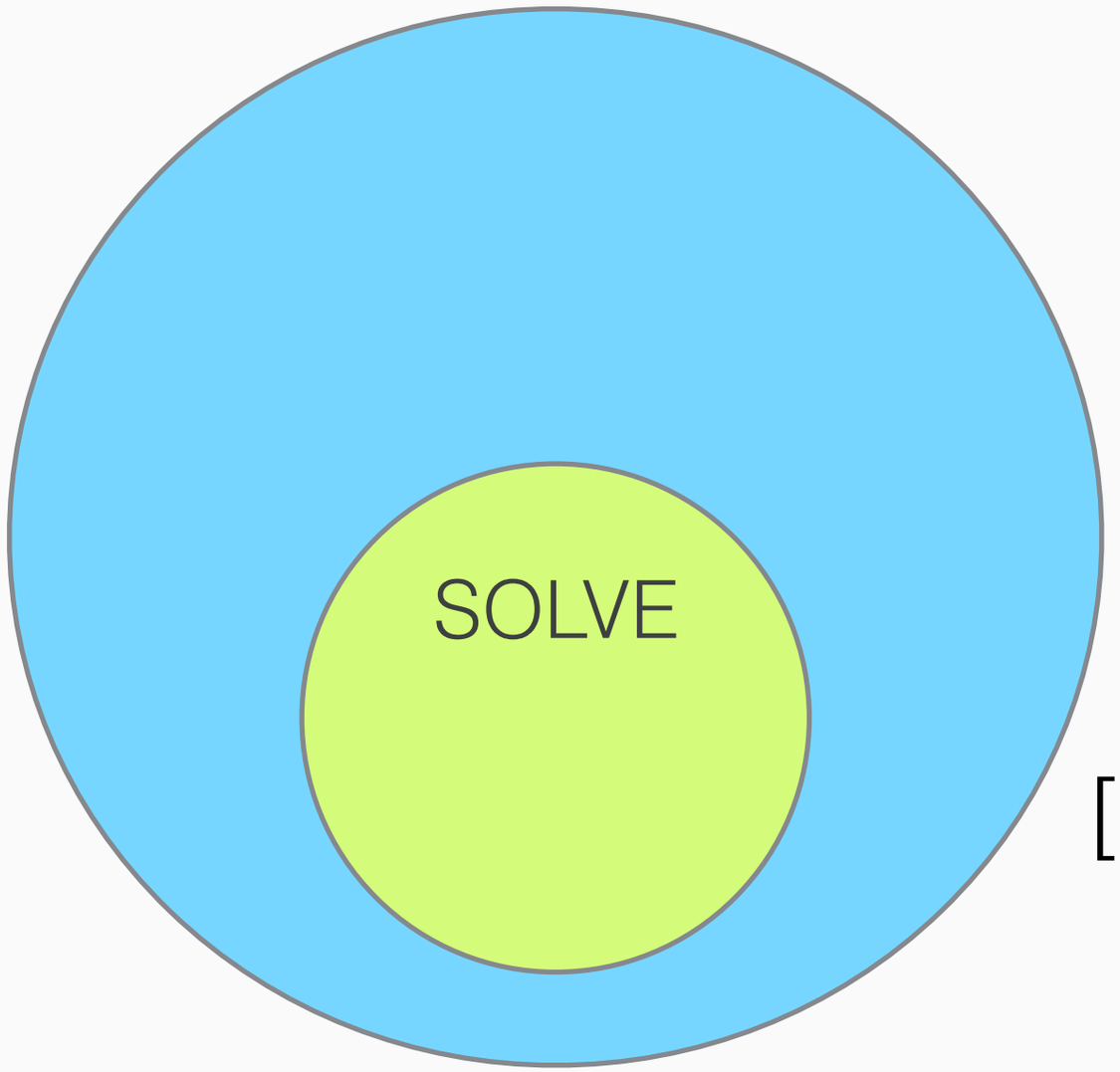
[A] Yes, we know it's in VERIFY

[B] Yes, we know it's not in VERIFY

[C] Nope!

[D] I'm confused!

Clicker Hint!



Q: Given *any* problem known to be in SOLVE, do we know anything about its status relative to VERIFY*?*

[A] Yes, we know it's in VERIFY

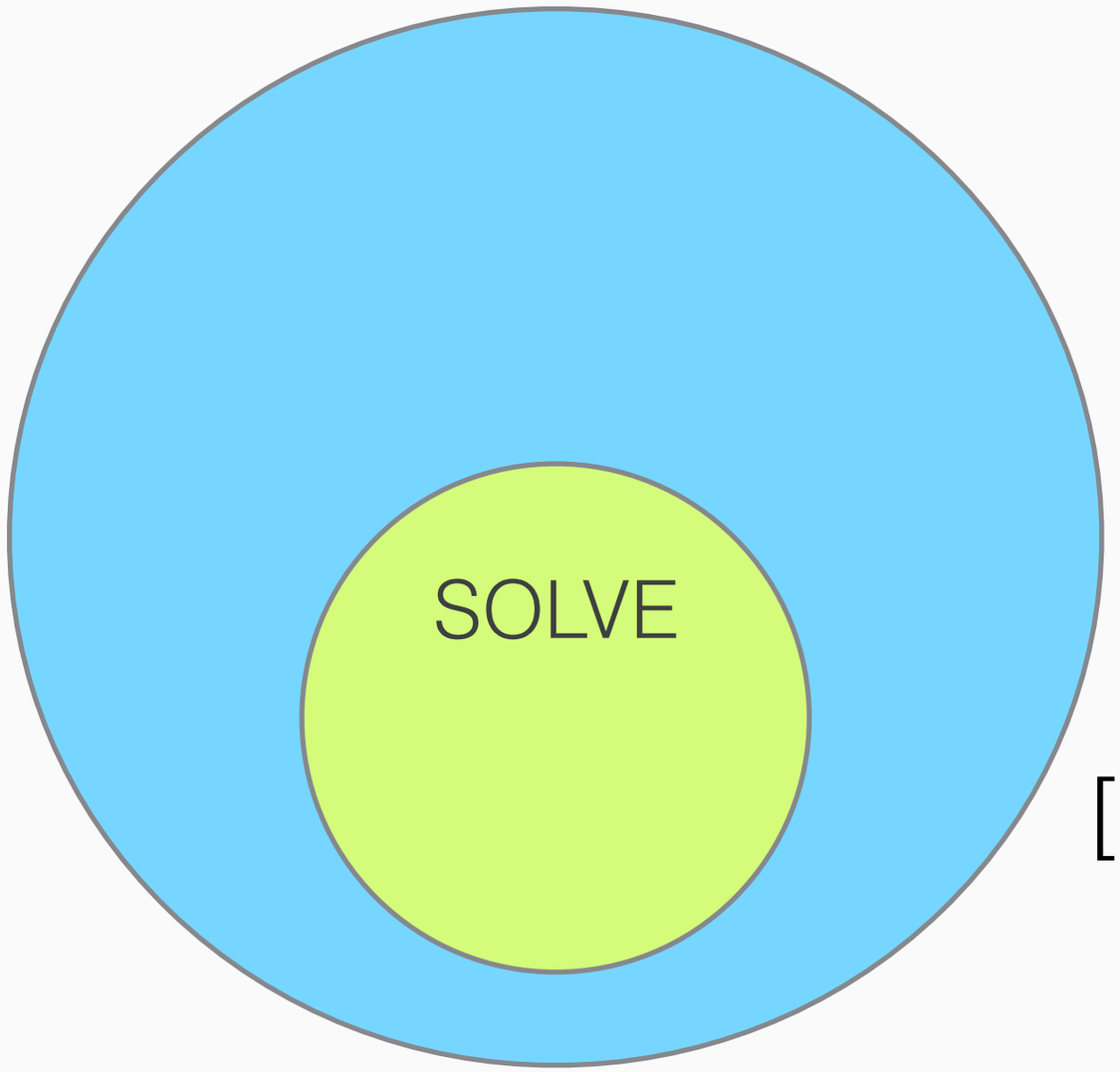
[B] Yes, we know it's not in VERIFY

[C] Nope!

[D] I'm confused!

(Think about Sorting)

Clicker Answer!



Q: Given *any* problem known to be in SOLVE, do we know anything about its status relative to VERIFY*?*

[A] Yes, we know it's in VERIFY

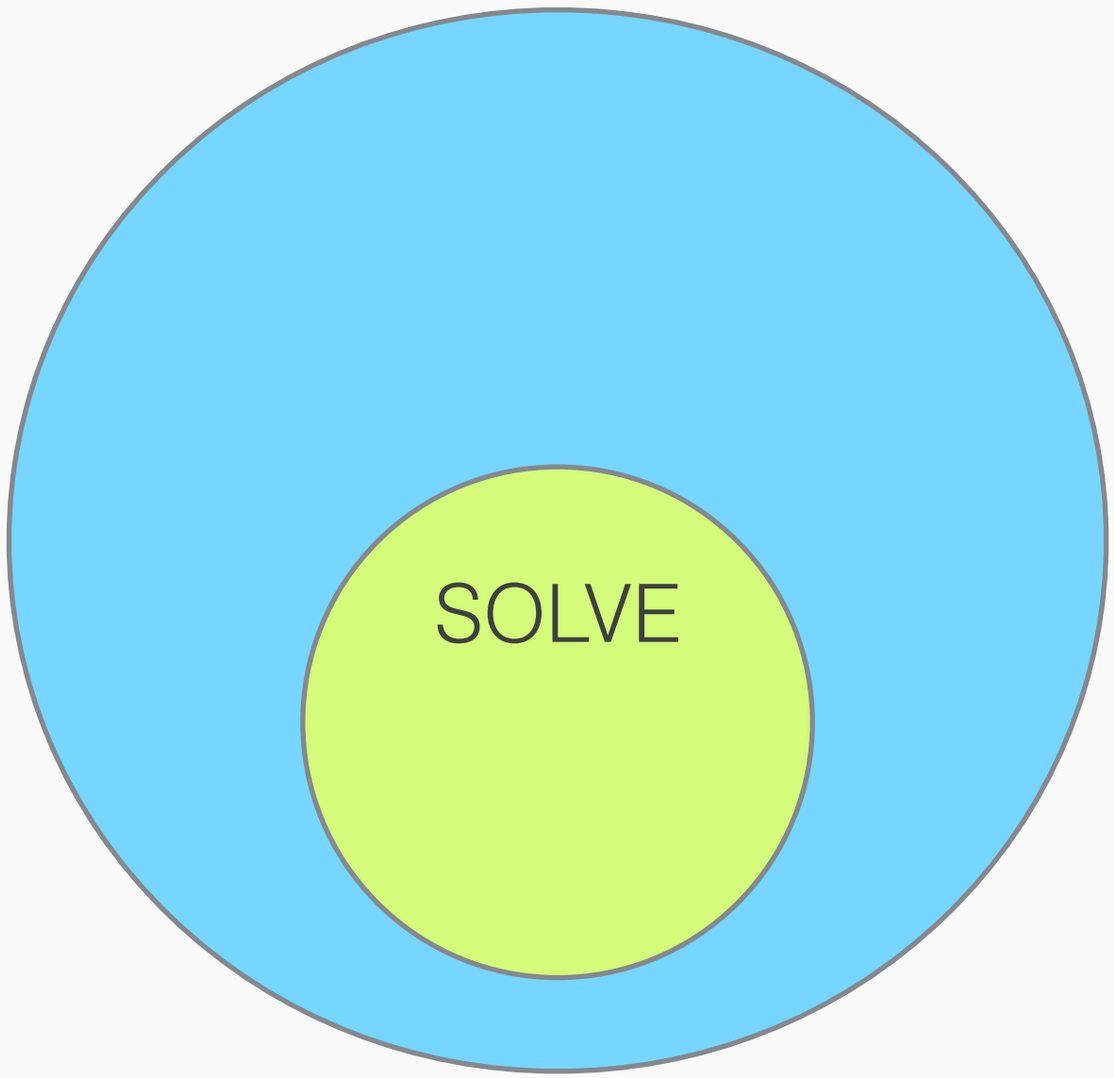
[B] Yes, we know it's not in VERIFY

[C] Nope!

[D] I'm confused!

(Think about Sorting)

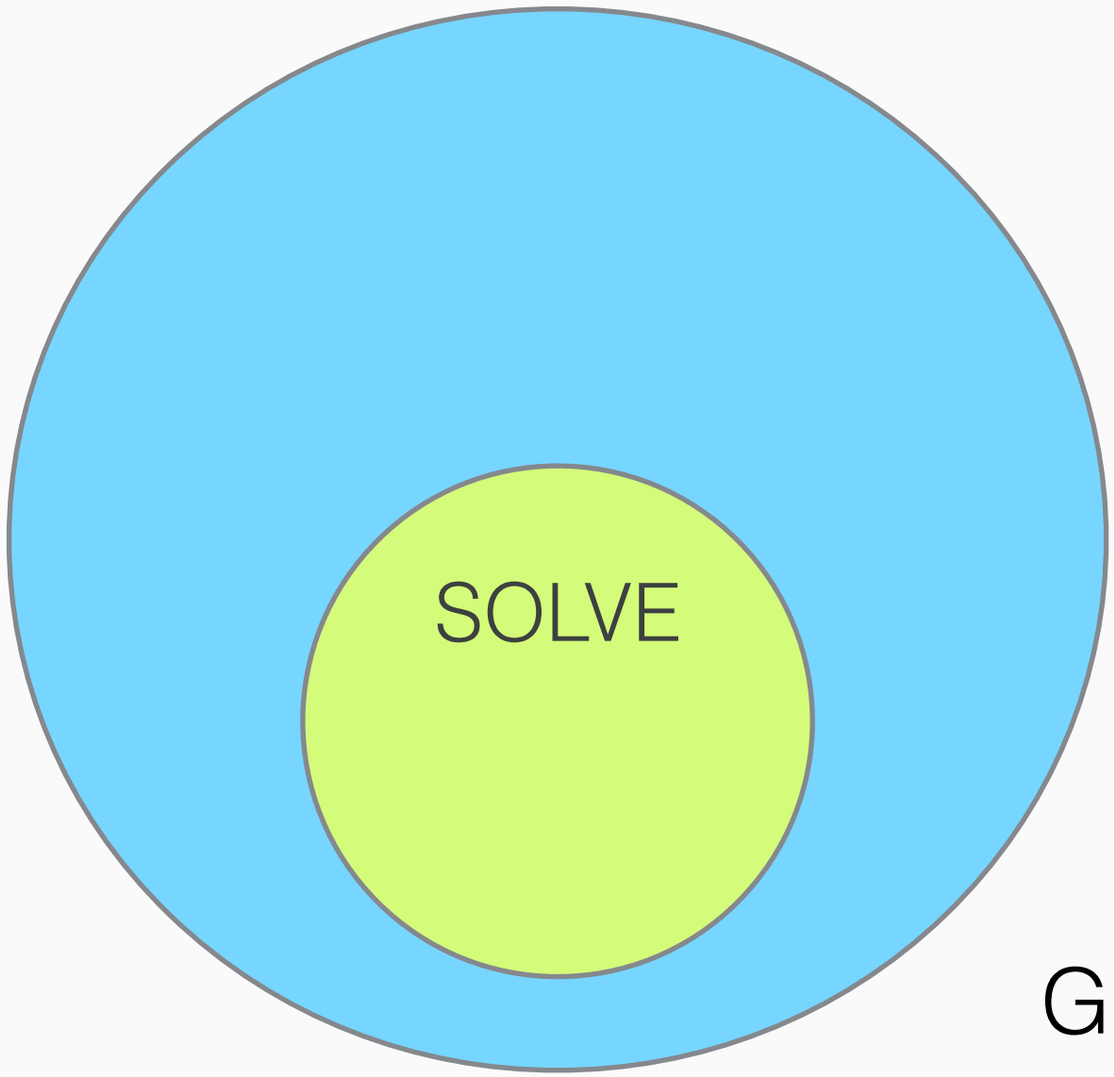
Neat Observation!



Any Problem in SOLVE is in VERIFY

Because you can always just solve the problem, then check to see if its the solution you were asked to verify!

Neat Observation!

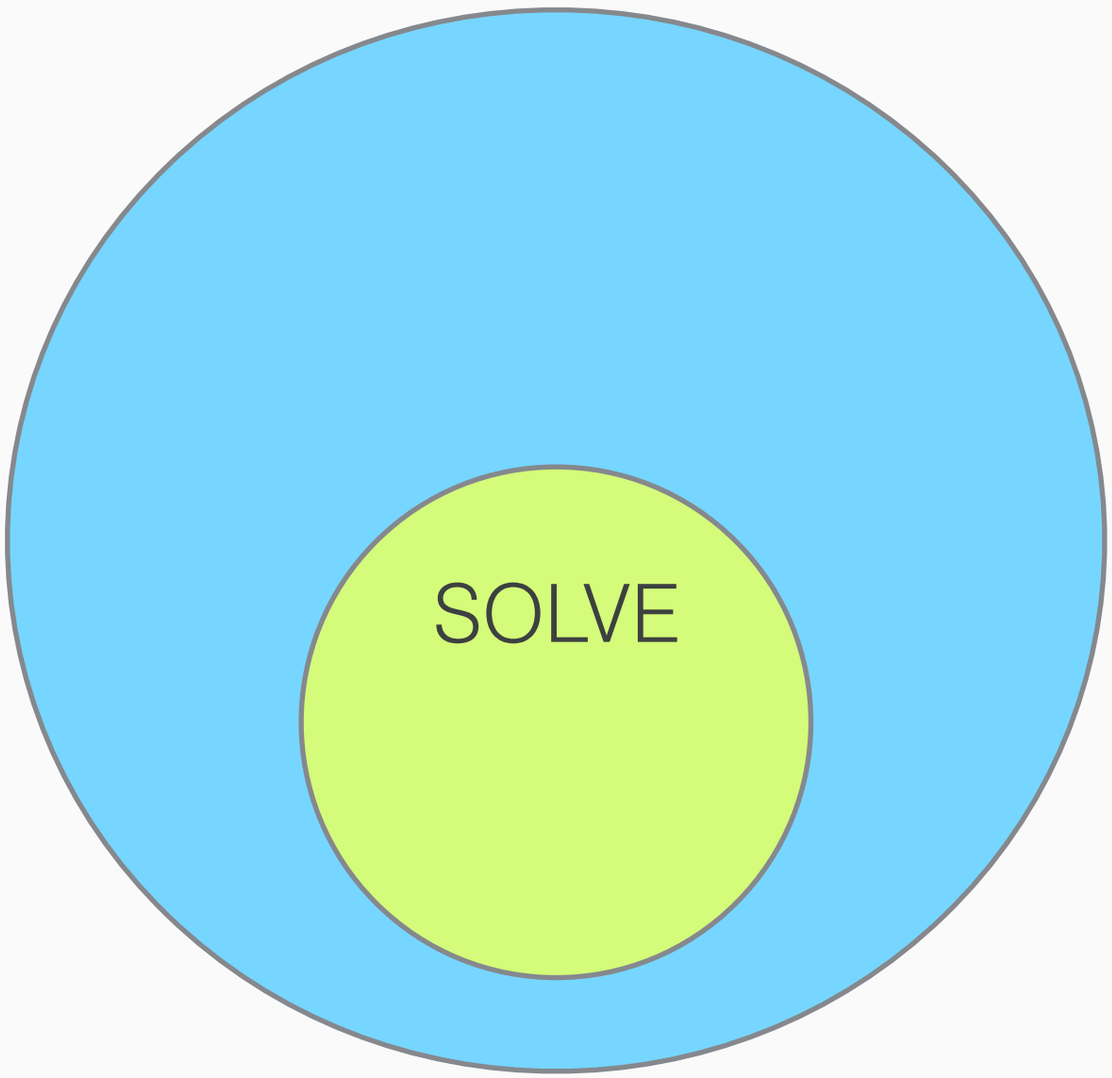


Any Problem in SOLVE is in VERIFY

Example: Sorting

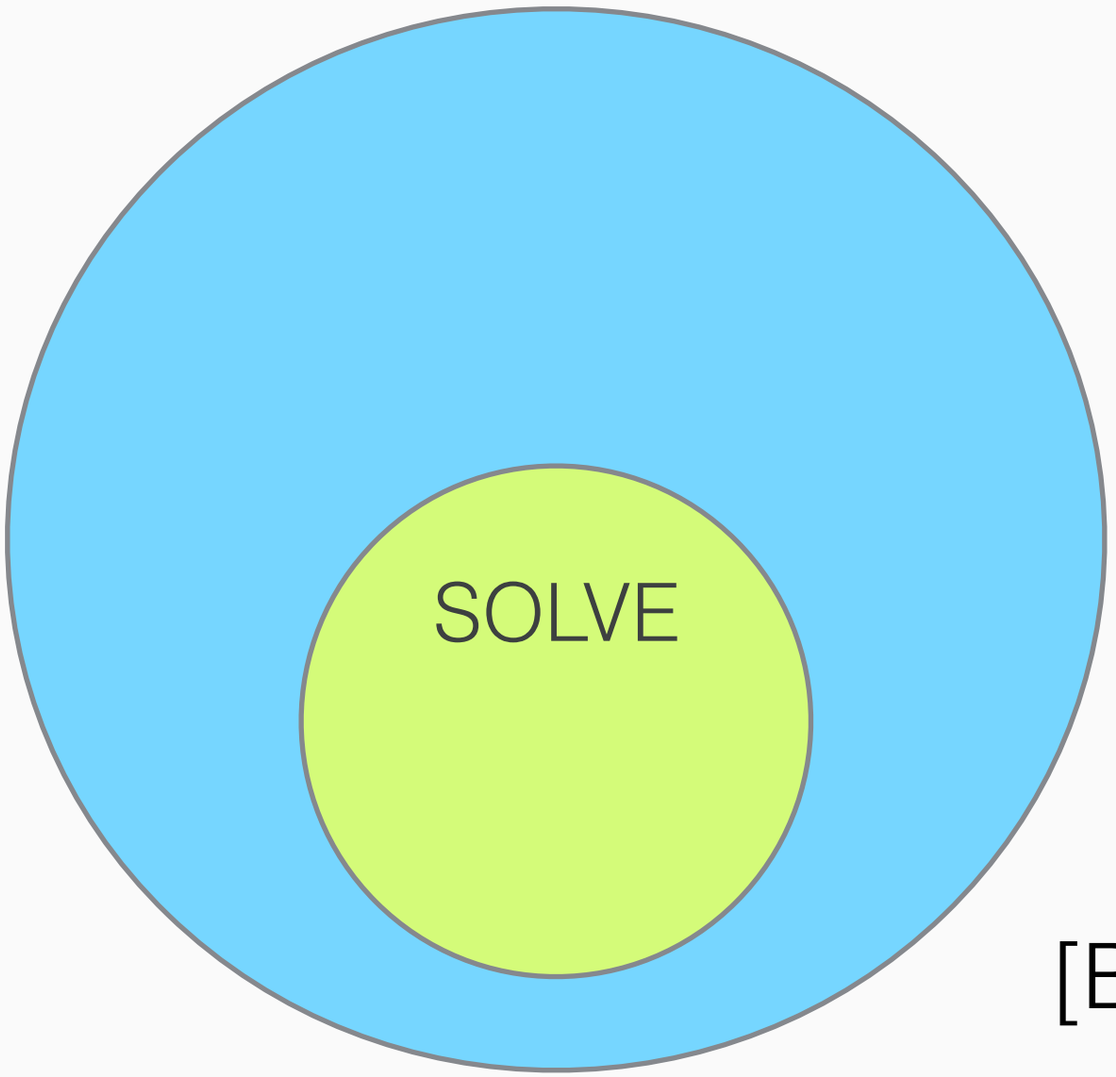
Given a sorting, run Selection Sort and then compare answers.

Clicker Question!



Q: Given *any* problem known to be in VERIFY, do we know anything about its status relative to SOLVE?

Clicker Question!



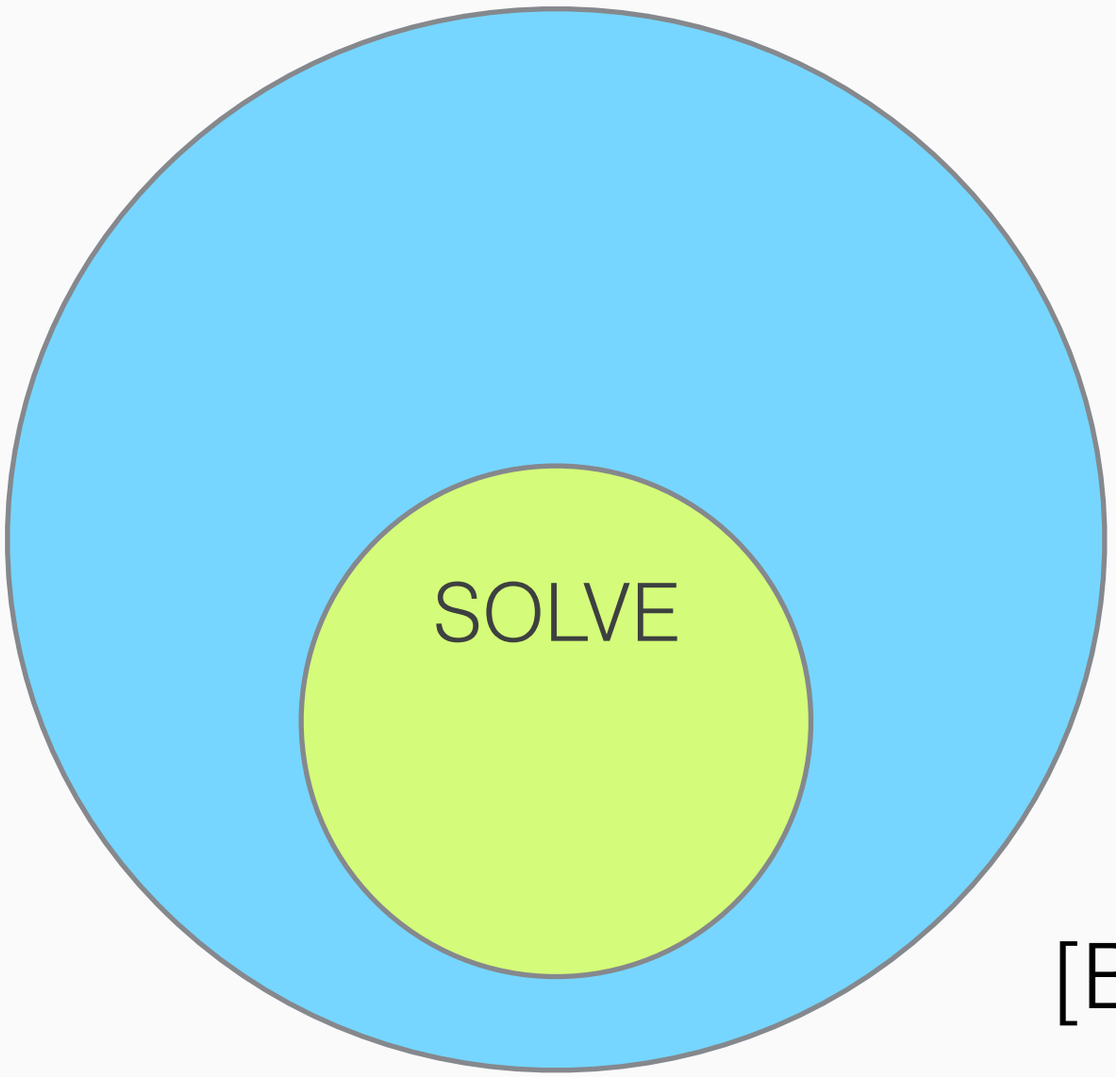
Q: Given *any* problem known to be in VERIFY, do we know anything about its status relative to SOLVE?

[A] Yes, we know it's in SOLVE

[B] Yes, we know it's not in SOLVE

[C] Nope!

Clicker ANSWER!



Q: Given *any* problem known to be in VERIFY, do we know anything about its status relative to SOLVE?

[A] Yes, we know it's in SOLVE

[B] Yes, we know it's not in SOLVE

[C] Nope! (so far)

This Is It!

Q: Given *any* problem known to be in VERIFY, do we know anything about its status relative to SOLVE?

A: Nope! (so far)

This is considered the most important unanswered question in all of computer science.

Rephrased

Q: If a problem's solution can be verified efficiently, can it also be solved efficiently?

This is considered the most important unanswered question in all of computer science.

Rephrased

Q: If a problem's solution can be verified efficiently, can it also be solved efficiently?

(for any problem we can think of)

This is considered the most important unanswered question in all of computer science.

Rephrased

Q: If a problem's solution can be verified efficiently, can it also be solved efficiently?

Observation: We could try guessing every answer and using our efficient verifier to verify it.

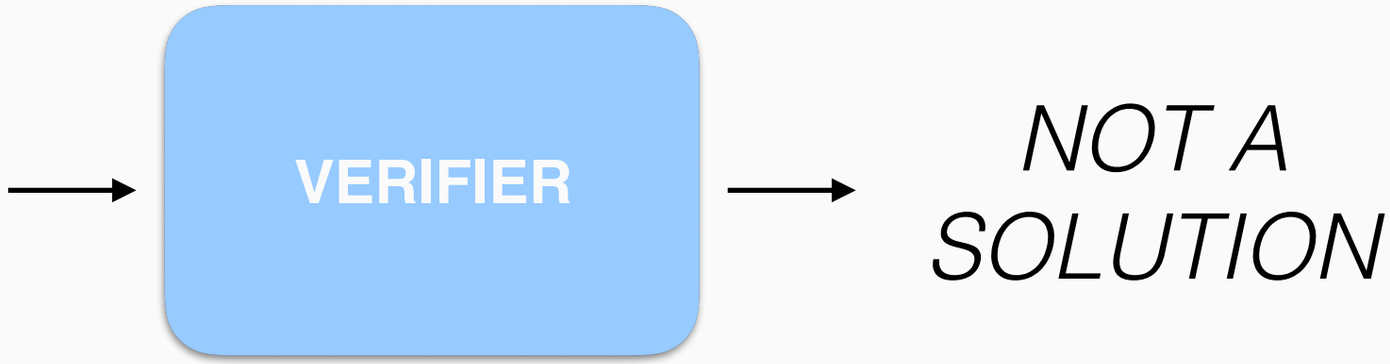
This is considered the most important unanswered question in all of computer science.

Rephrased

5	3	0	0	7	0	0	0	0
6	0	0	1	9	5	0	0	0
0	9	8	0	0	0	0	6	0
8	0	0	0	6	0	0	0	3
4	0	0	8	0	3	0	0	1
7	0	0	0	2	0	0	0	6
0	6	0	0	0	0	2	8	0
0	0	0	4	1	9	0	0	5
0	0	0	0	8	0	0	7	9

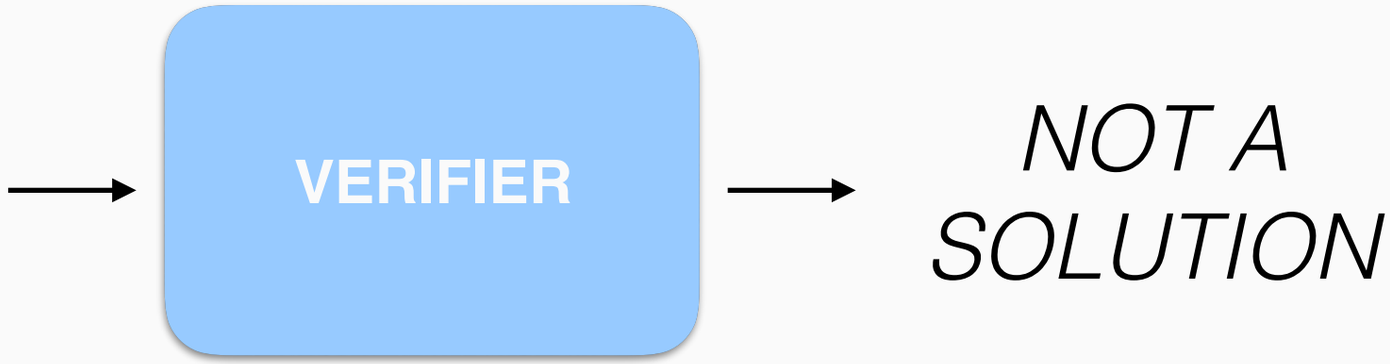
Observation: We could try guessing every answer and using our efficient verifier to verify it.

Rephrased



Observation: We could try guessing every answer and using our efficient verifier to verify it.

Rephrased



Observation: We could try guessing every answer and using our efficient verifier to verify it.

But what about problems for which there are too many possible answers? (e.g Chess, Go, etc.)

Some Terminology

SOLVE is commonly called “P” for polynomial

VERIFY is commonly called “NP”, for non-deterministic polynomial.

(but don’t worry about the names!)

Some Terminology

SOLVE is commonly called “P” for polynomial

VERIFY is commonly called “NP”, for non-deterministic polynomial

The question is commonly called “P versus NP”

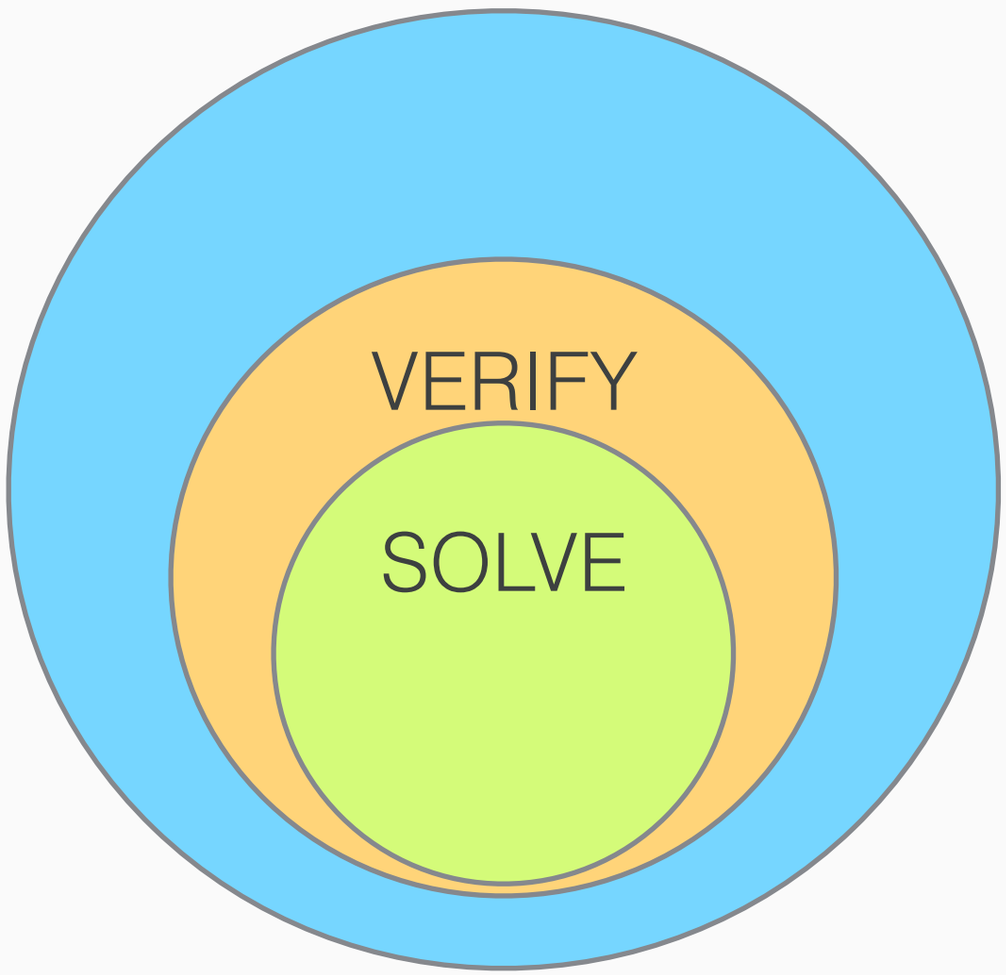
P versus NP

- One of seven problems pitched by the Clay Mathematics Institute in the year 2000 as the *most important unsolved mathematical questions*.
- Solve one? Get $1,000,000.
- Only one has been solved so far (not P vs. NP).

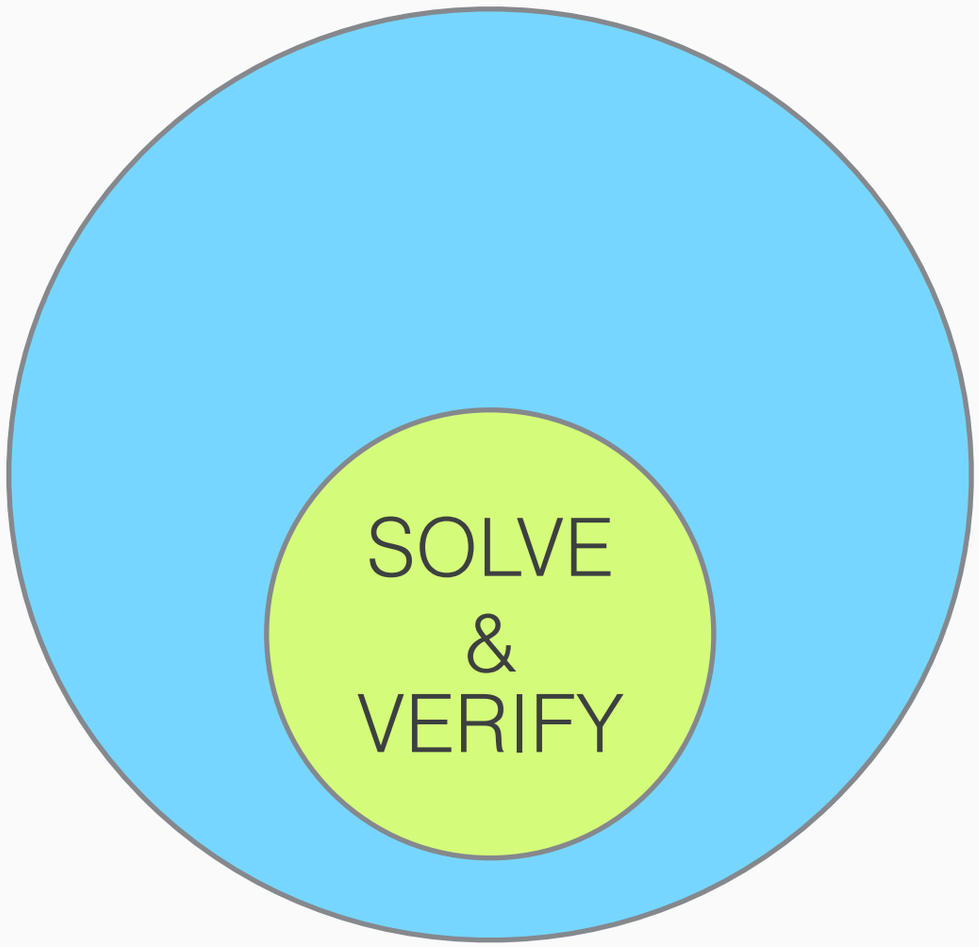
The question is commonly called "P versus NP"

SOLVE versus VERIFY

A: No!



A: Yes!



Q: If a problem's solution can be verified efficiently, can it also be solved efficiently?

P versus NP

Reflection

- We can think about problem classes in terms of how fast the *fastest possible algorithm* for the problem is.
- One class of interest is SOLVE, the set of problems we can solve efficiently (before sun goes poof)!
- Another class of interest is VERIFY, the set of problems whose solutions we can verify efficiently!
- If a problem is in SOLVE, we know it's in VERIFY.
- Q: If a problem is in VERIFY, is it also in solve? Is the most important unanswered question in computer science.

Up Next

- Some Implications
- We'll look at some famous example problems that are in VERIFY.
- Revisiting The Halting Problem, more discussions about the unsolvable.

Implications

Implications: VERIFY = SOLVE

- Then anytime a problem has a method for *verifying* a solution efficiently, there's also a method for *finding* a solution efficiently.
- Here's a *totally insane result* if that's true:
  - We already have an efficient method for verifying mathematical proofs.
  - Therefore, there is an efficient method for finding mathematical proofs of arbitrary statements.

Implications: VERIFY = SOLVE

- Then anytime a problem has a method for *verifying* a solution efficiently, there's also a method for *finding* a solution efficiently.

- Here's a *totally insane result* if that's true:

  - We have an efficient method for verifying mathematical proofs.

  - Therefore, there is an efficient method for finding mathematical proofs of arbitrary statements.

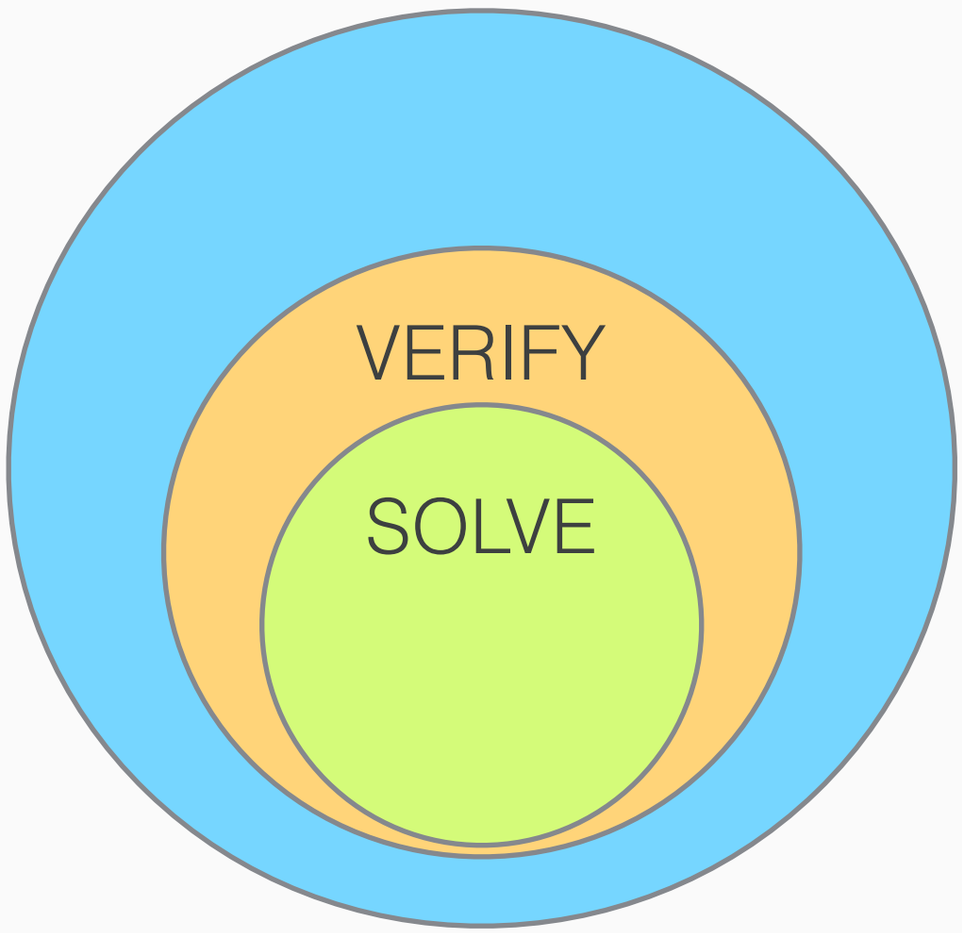
  - **That means we can solve every other millennium problem!**

Implications: VERIFY = SOLVE

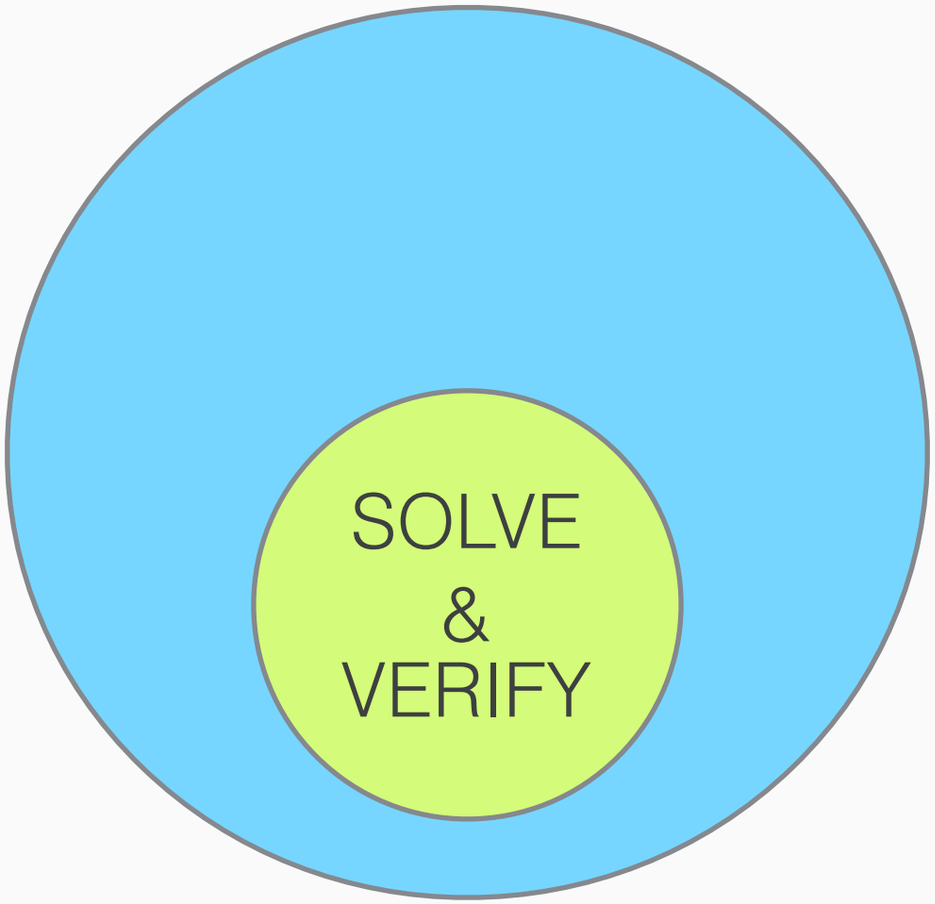
- Then anytime a problem has a method for *verifying* a solution efficiently, there's also a method for *finding* a solution efficiently.
- Here's **another** *totally insane result* if that's true:
  - The security systems we currently rely on become *useless* overnight.
  - We'll talk more in Cryptography about this!

So, does VERIFY = SOLVE?

A: No!



A: Yes!



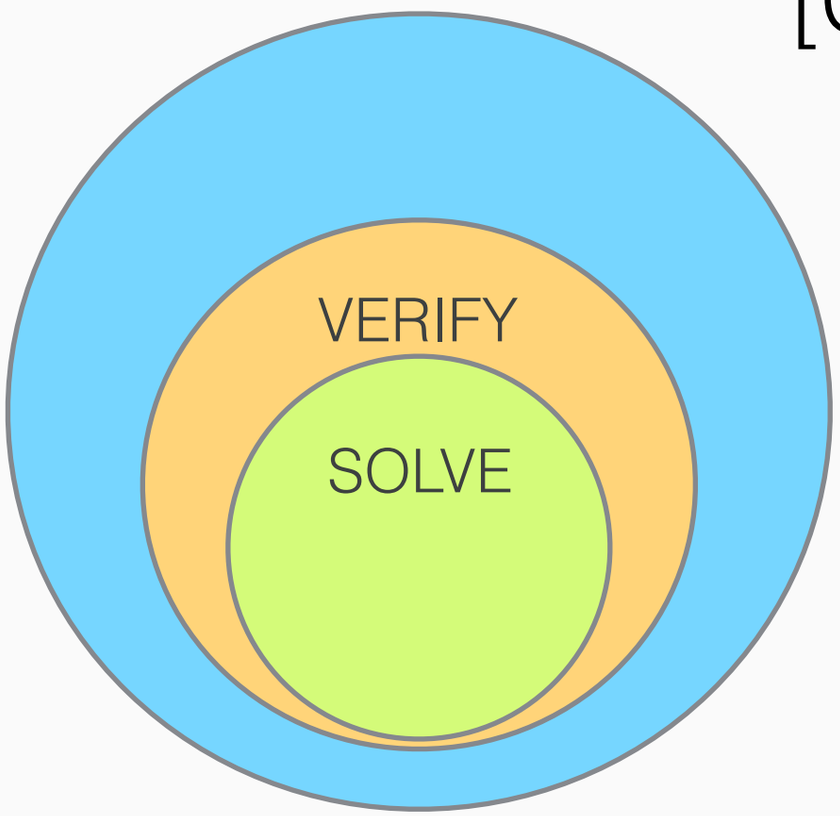
Q: If a problem's solution can be verified efficiently, can it also be solved efficiently?

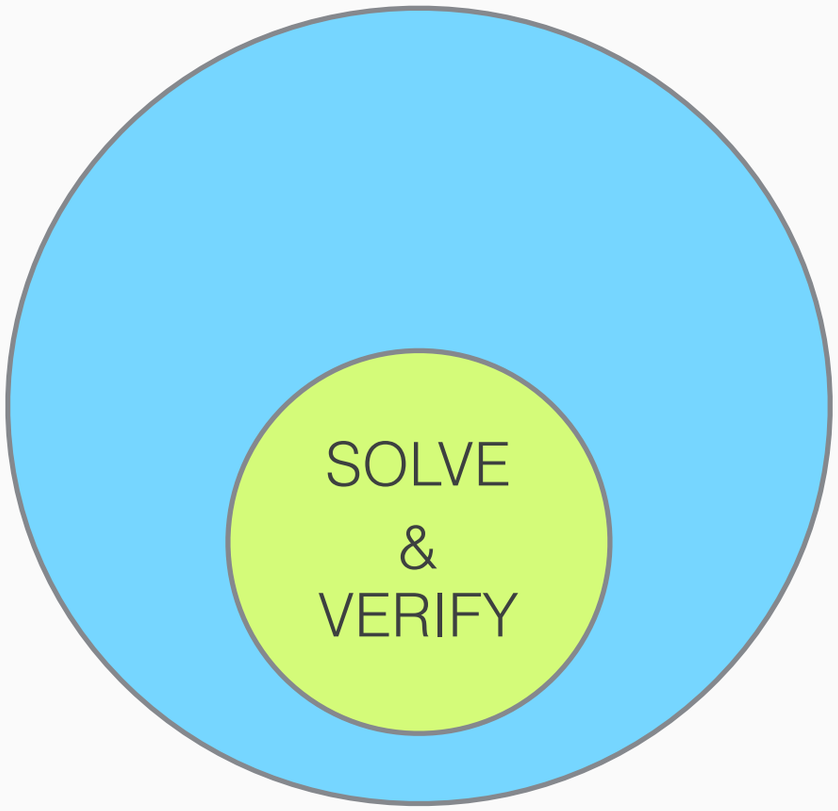
Clicker Question!

[A]: No!

[B]: Yes!

[C]: I'm confused

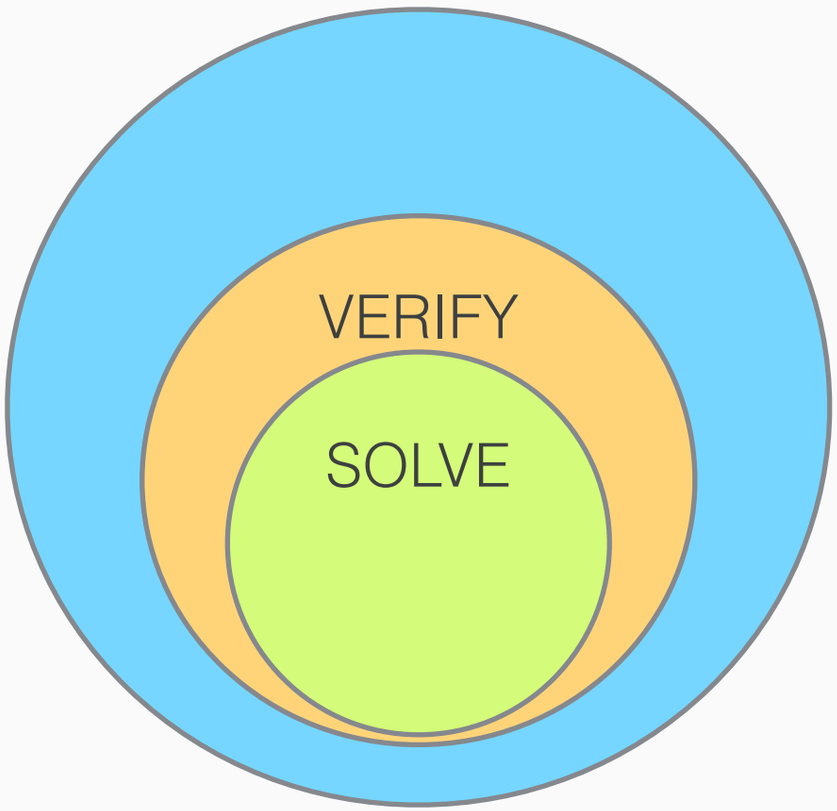




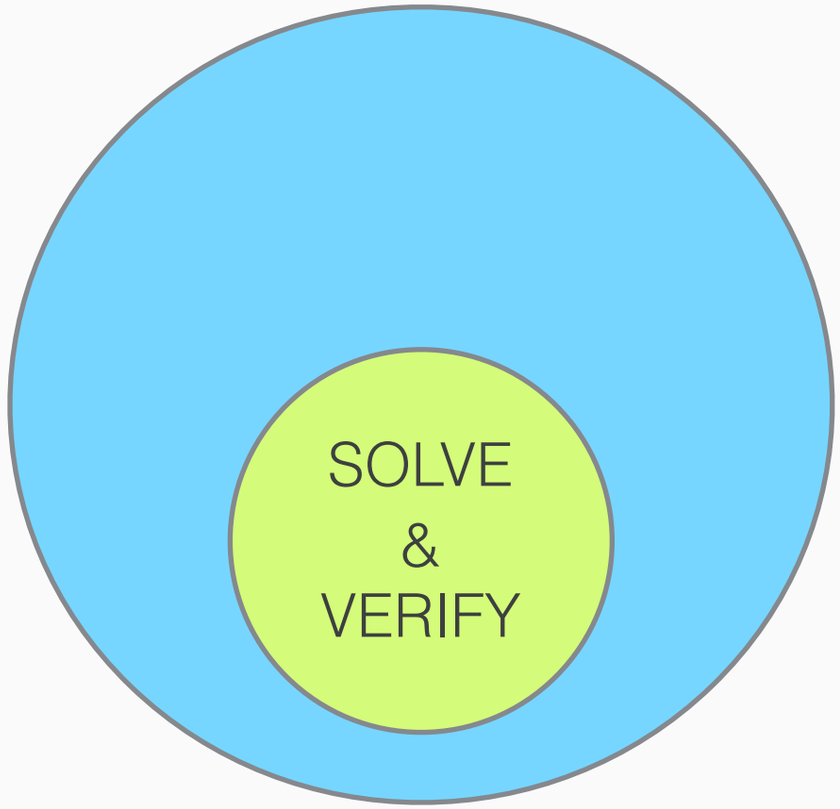
Q: What do you think?

So, does VERIFY = SOLVE?

A: No!



A: Yes!



Most computer scientists think the answer is “No”