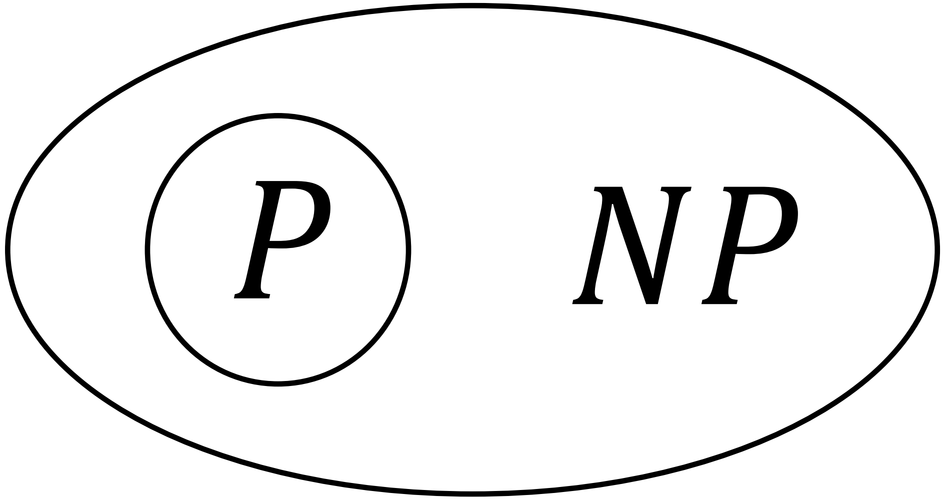
NP-Complete

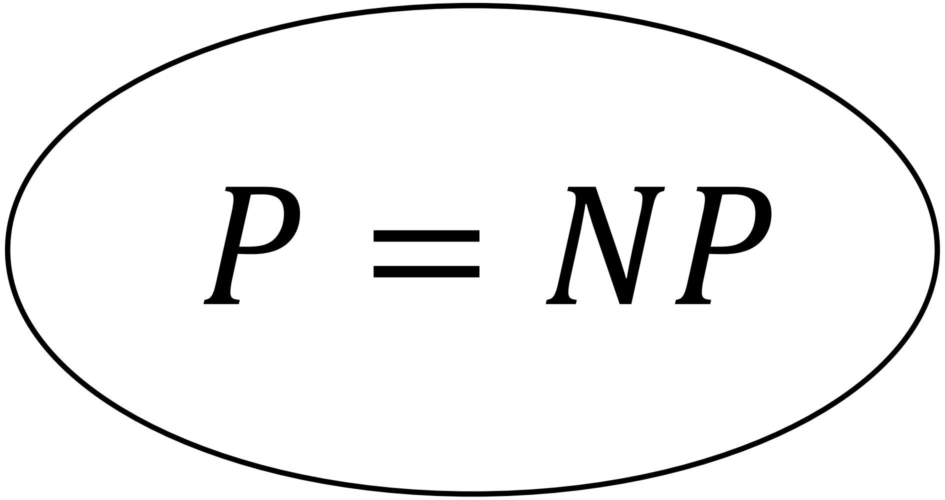
CSCI 338

P versus NP

History Lesson:

- **P vs NP concepts first discussed in 1950's**
- **P vs NP formalized in 1971**





Can all problems that are verifiable in polynomial time be solved in polynomial time?

SAT & $3SAT$

$$\phi = (x_1 \vee x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_2)$$

ϕ is a formula with clauses composed of Boolean variables connected by ORs, and clauses connected by ANDs.

SAT & $3SAT$

$$\phi = (x_1 \vee x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_2)$$

ϕ is a formula with clauses composed of Boolean variables connected by ORs, and clauses connected by ANDs.

SAT & $3SAT$

$$\phi = (x_1 \vee x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_2)$$

ϕ is a formula with clauses composed of Boolean variables connected by ORs, and clauses connected by ANDs.

SAT & $3SAT$

$$\phi = (x_1 \lor x_1 \lor x_2) \land (\overline{x_1} \lor \overline{x_2} \lor \overline{x_2}) \land (\overline{x_1} \lor x_2 \lor x_2)$$

ϕ is a formula with clauses composed of Boolean variables connected by ORs, and clauses connected by ANDs.

SAT & $3SAT$

$$\phi = (x_1 \lor x_1 \lor x_2) \land (\overline{x_1} \lor \overline{x_2} \lor \overline{x_2}) \land (\overline{x_1} \lor x_2 \lor x_2)$$

ϕ is a formula with clauses composed of Boolean variables connected by ORs, and clauses connected by ANDs.

(called conjunctive normal form – CNF)

SAT & $3SAT$

$$\phi = (x_1 \vee x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_2)$$

ϕ is a formula with clauses composed of Boolean variables connected by ORs, and clauses connected by ANDs.

Can you set the variables to **true** or **false** so that ϕ evaluates to **true**?

$SAT \ \& \ 3SAT$

$$\phi = (x_1 \lor x_1 \lor x_2) \land (\overline{x_1} \lor \overline{x_2} \lor \overline{x_2}) \land (\overline{x_1} \lor x_2 \lor x_2)$$

$x_1 = false$
$x_2 = true$

SAT & $3SAT$

$$\phi = (x_1 \vee x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_2)$$

$$(F \vee F \vee T) \qquad (T \vee F \vee F) \qquad (T \vee T \vee T)$$

$x_1 = false$
$x_2 = true$

$SAT \& 3SAT$

$$\phi = (x_1 \vee x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_2)$$

$$(F \vee F \vee T) \qquad (T \vee F \vee F) \qquad (T \vee T \vee T)$$

$$T \qquad T \qquad T$$

$x_1 = false$

$x_2 = true$

SAT & $3SAT$

$$\phi = (x_1 \vee x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_2)$$

$$\begin{array}{ccc} \downarrow & \downarrow & \downarrow \\ (F \vee F \vee T) & (T \vee F \vee F) & (T \vee T \vee T) \\ \downarrow & \downarrow & \downarrow \\ T & T & T \\ \searrow & \downarrow & \swarrow \\ & T & \end{array}$$

$x_1 = false$

$x_2 = true$

SAT & $3SAT$

$$\phi = (x_1 \lor x_1 \lor x_2) \land (\overline{x_1} \lor \overline{x_2} \lor \overline{x_2}) \land (\overline{x_1} \lor x_2 \lor x_2)$$

$$(F \lor F \lor T) \qquad (T \lor F \lor F) \qquad (T \lor T \lor T)$$

$$T \qquad T \qquad T$$

$x_1 = false$

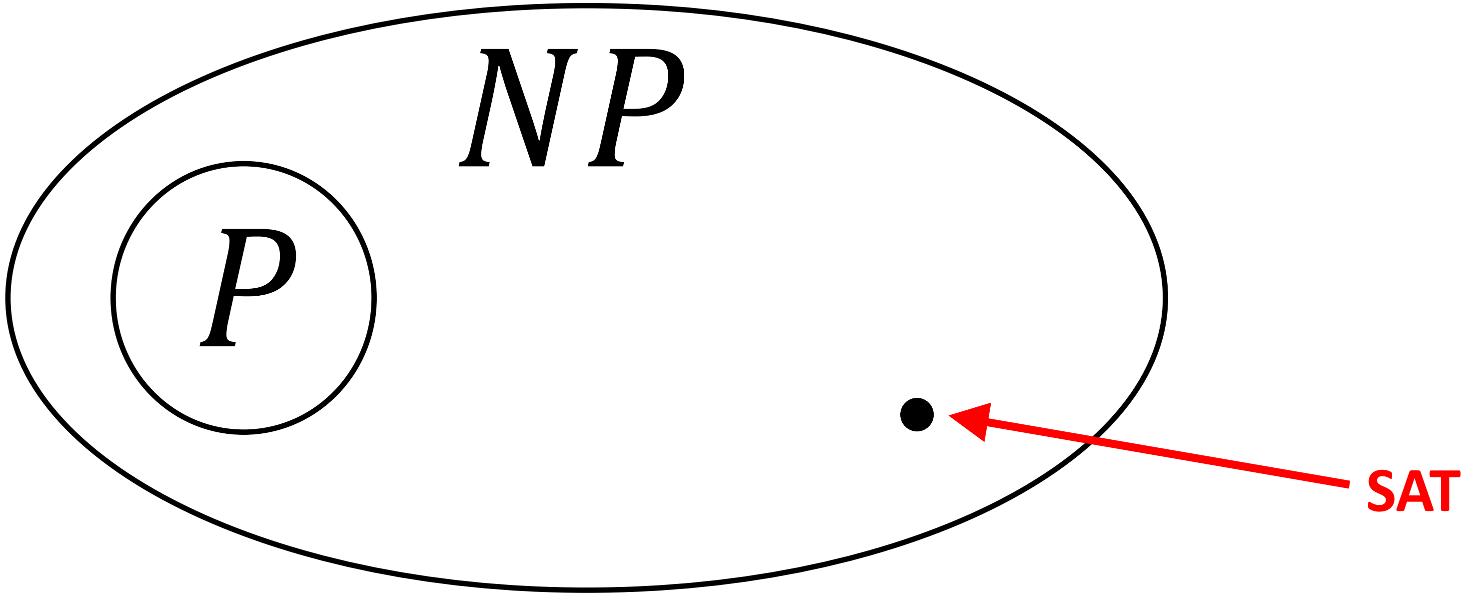
$x_2 = true$

$$T$$

$SAT = \{\langle\phi\rangle\colon$ ϕ is a satisfiable formula$\}$

$3SAT = \{\langle\phi\rangle\colon$ ϕ is a satisfiable formula with 3 variables per clause$\}$

P versus NP



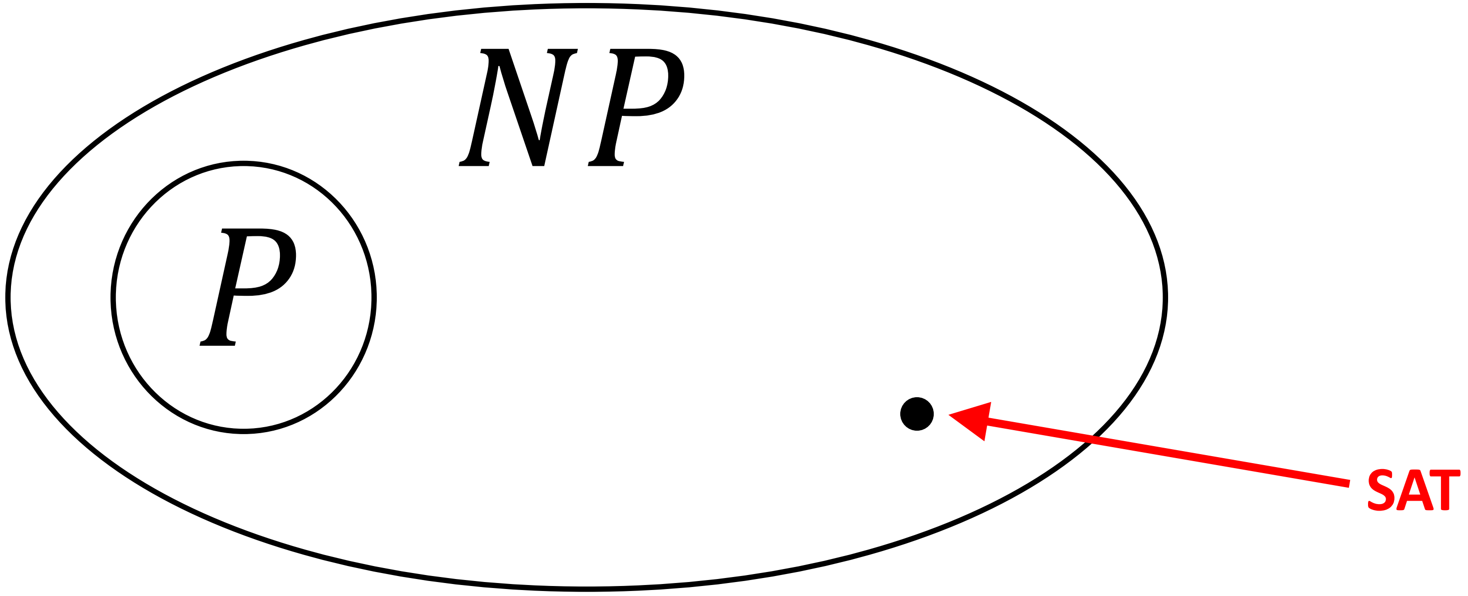
History Lesson:

- P vs NP concepts first discussed in 1950's
- P vs NP formalized in 1971
- SAT proven to solve everything in NP in 1971

Cook-Levin Theorem: Every problem in NP can be solved by a solver for SAT with at most polynomial extra time.

Proof:

P versus NP

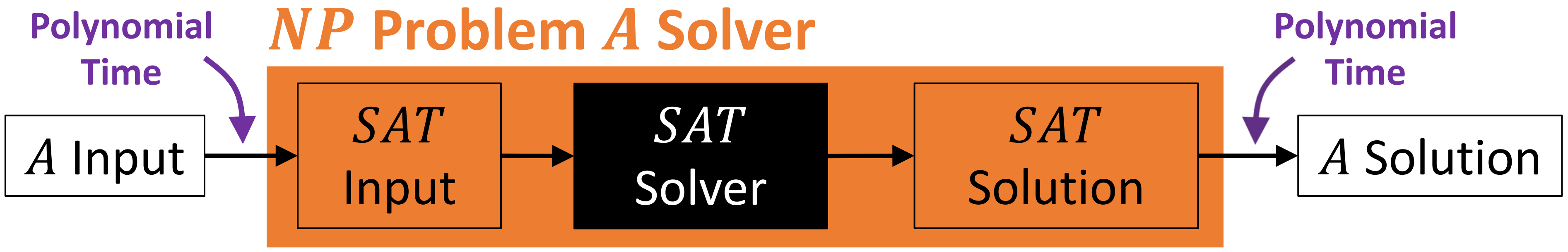


History Lesson:

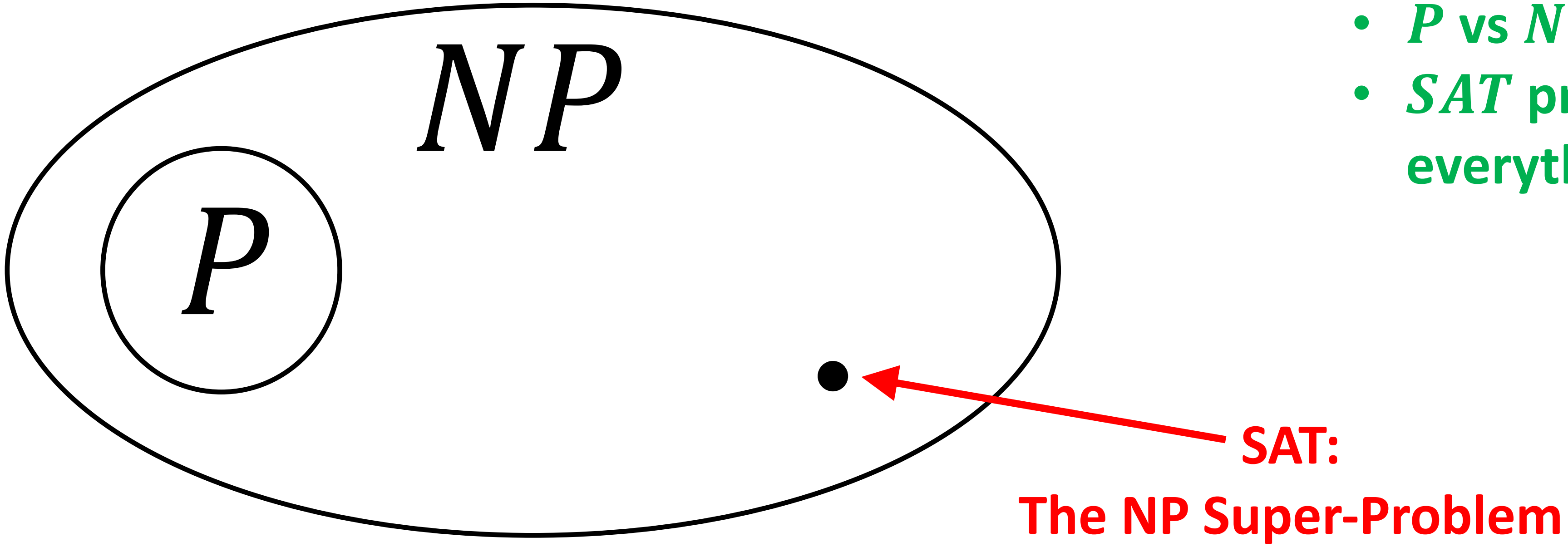
- P vs NP concepts first discussed in 1950's
- P vs NP formalized in 1971
- SAT proven to solve everything in NP in 1971

Cook-Levin Theorem: Every problem in NP can be solved by a solver for SAT with at most polynomial extra time.

Proof:



P versus NP

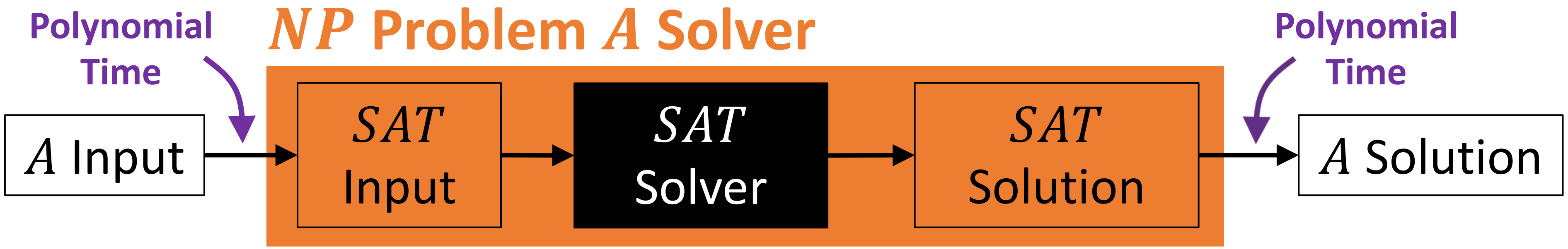


History Lesson:

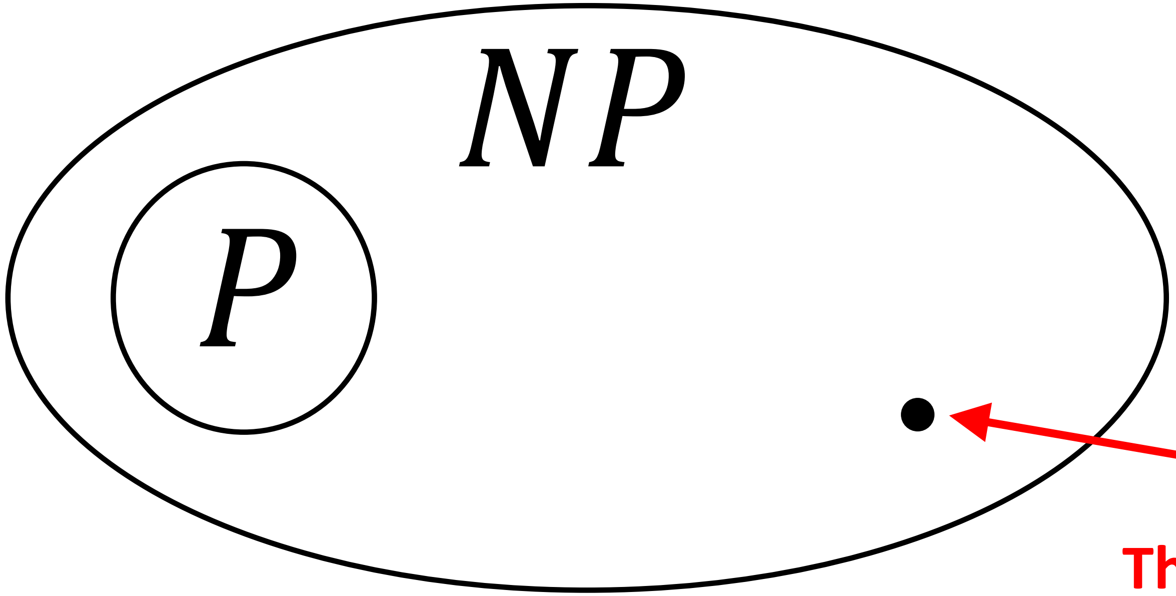
- P vs NP concepts first discussed in 1950's
- P vs NP formalized in 1971
- SAT proven to solve everything in NP in 1971

Cook-Levin Theorem: Every problem in NP can be solved by a solver for SAT with at most polynomial extra time.

Proof:



P versus NP



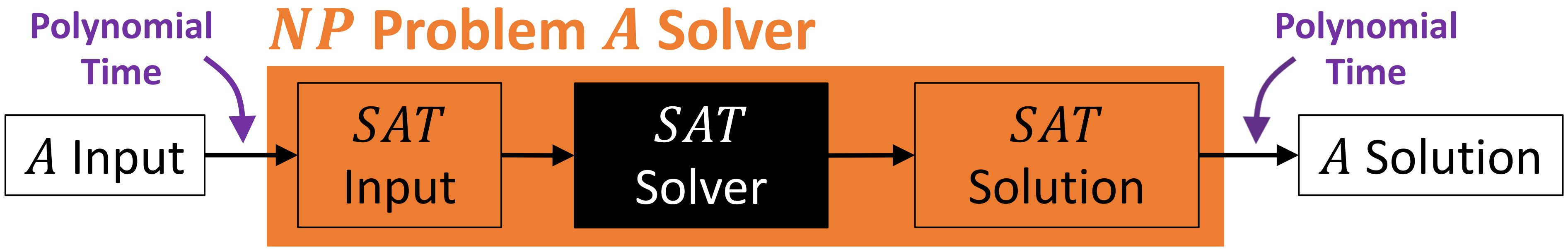
SAT:
The NP Super-Problem

History Lesson:
- P vs NP concepts first discussed in 1950's
- P vs NP formalized in 1971
- SAT proven to solve everything in NP in 1971

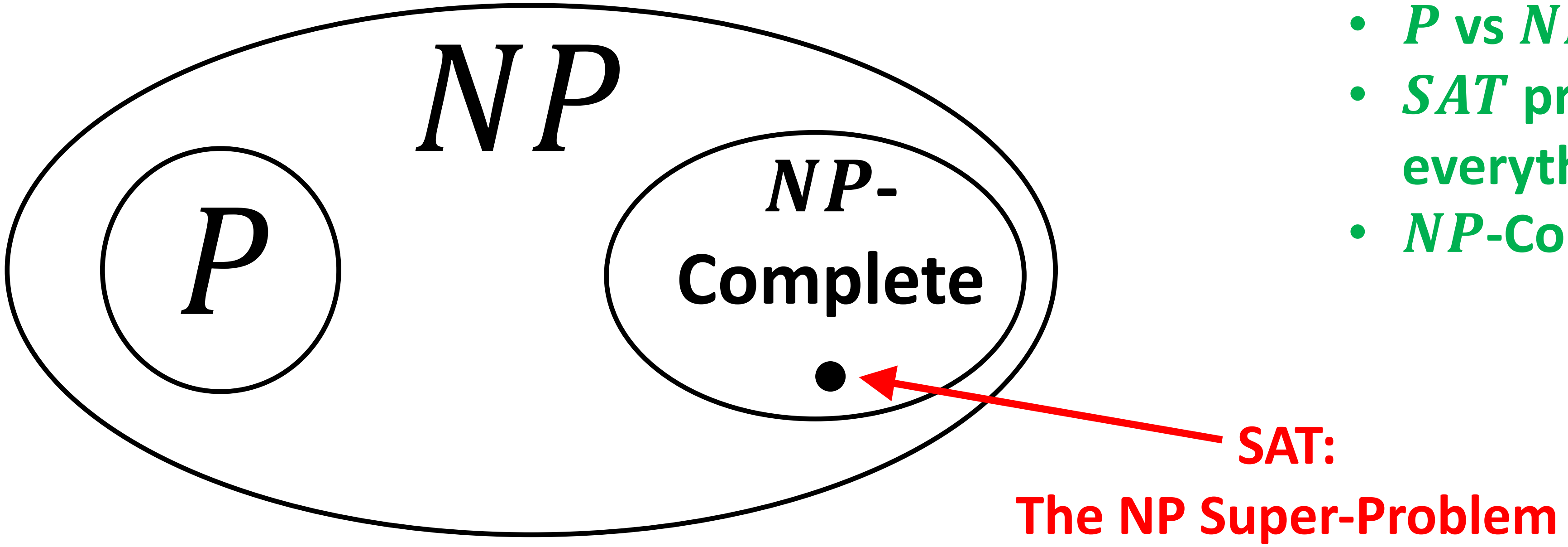
Cook-Levin Theorem: Every problem in NP can be solved by a solver for SAT with at most polynomial extra time.

Proof:

Complicated!



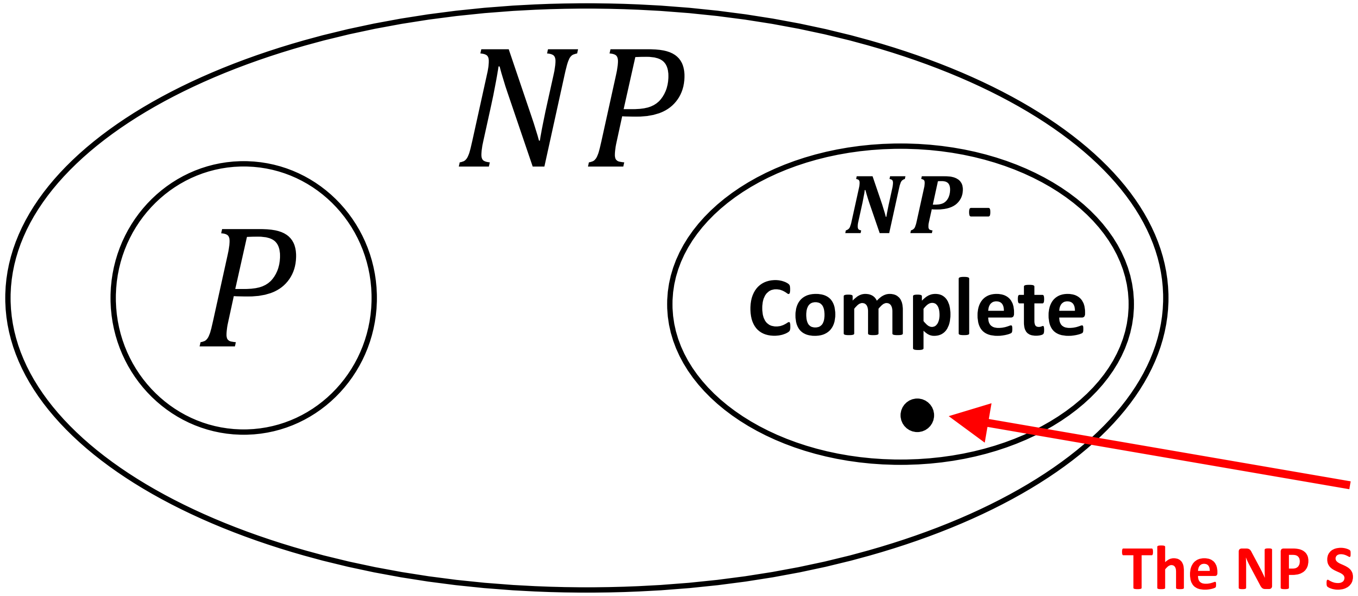
NP-Complete



History Lesson:

- P vs NP concepts first discussed in 1950's
- P vs NP formalized in 1971
- SAT proven to solve everything in NP in 1971
- NP-Complete defined in 1972

NP-Complete



SAT:
The NP Super-Problem

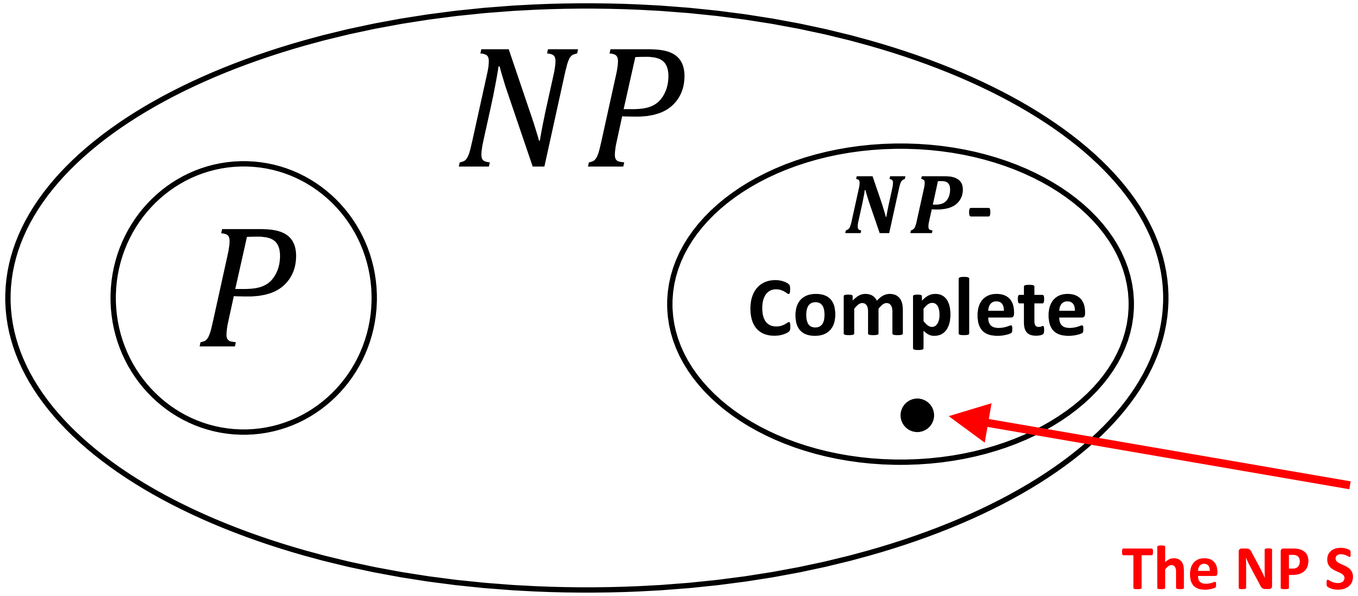
<u>**History Lesson:**</u>

- **P vs NP concepts first discussed in 1950's**
- **P vs NP formalized in 1971**
- **SAT proven to solve everything in NP in 1971**
- **NP-Complete defined in 1972**

B is in NP-Complete if it satisfies two conditions:

1. $B \in NP$.
2. For every $A \in NP$, $A \leq_P B$.

NP-Complete



History Lesson:

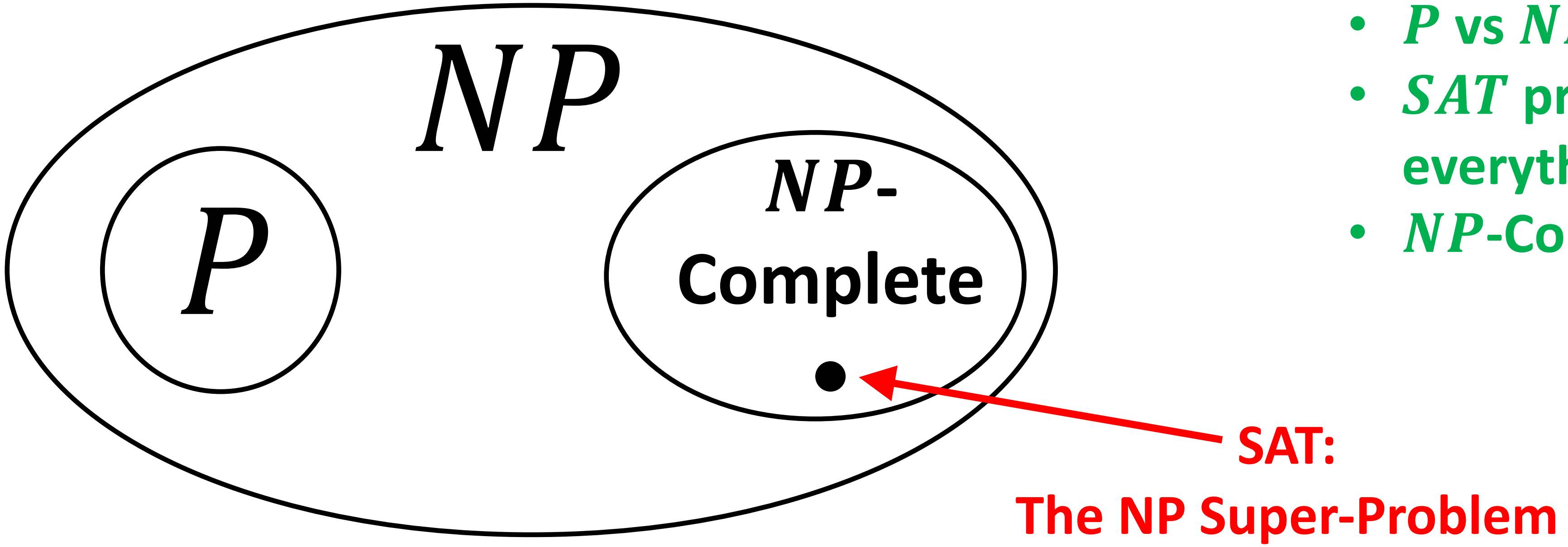
- P vs NP concepts first discussed in 1950's
- P vs NP formalized in 1971
- SAT proven to solve everything in NP in 1971
- NP-Complete defined in 1972

B is in NP-Complete if it satisfies two conditions:

1. $B \in NP$.
2. For every $A \in NP$, $A \leq_P B$.

"Every problem in NP can be solved by an algorithm for B in polynomial extra time."

NP-Complete



History Lesson:

- P vs NP concepts first discussed in 1950's
- P vs NP formalized in 1971
- SAT proven to solve everything in NP in 1971
- NP-Complete defined in 1972

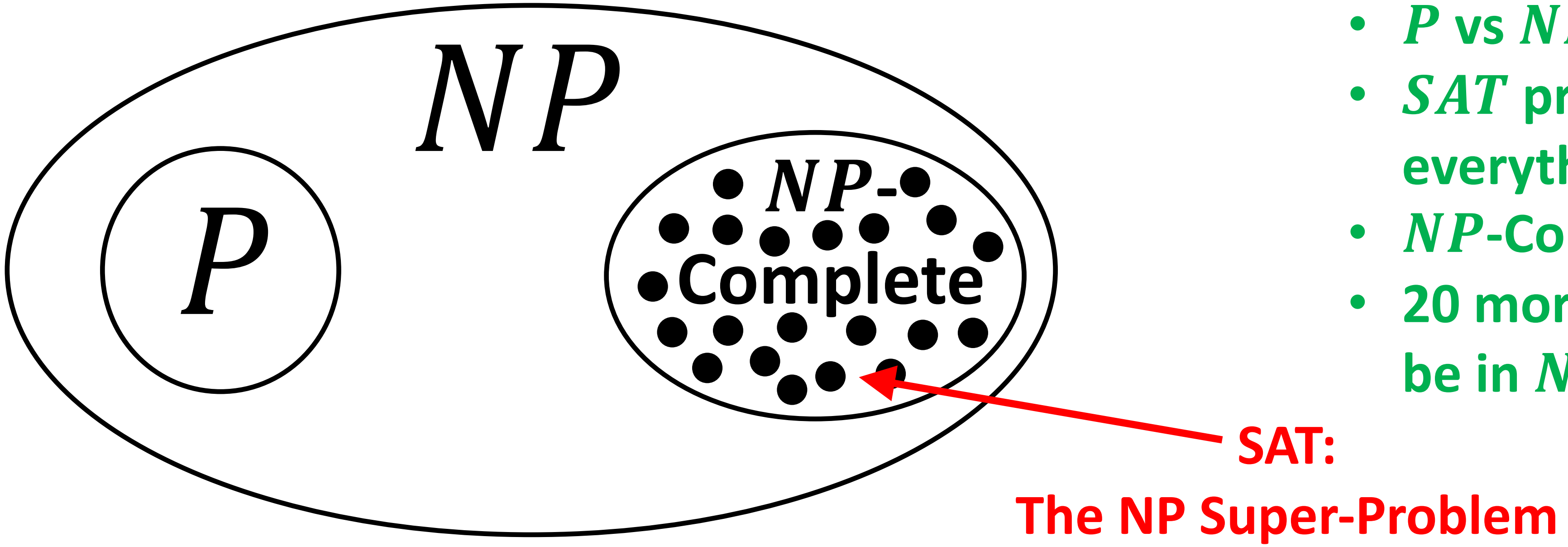
B is in NP-Complete if it satisfies two conditions:

1. $B \in NP$.
2. For every $A \in NP$, $A \leq_P B$.

B is in NP-Complete if it satisfies two conditions:

1. $B \in NP$.
2. For some $A \in NP$-C, $A \leq_P B$.

NP-Complete



History Lesson:

- P vs NP concepts first discussed in 1950's
- P vs NP formalized in 1971
- SAT proven to solve everything in NP in 1971
- NP-Complete defined in 1972
- 20 more problems shown to be in NP-Complete in 1972

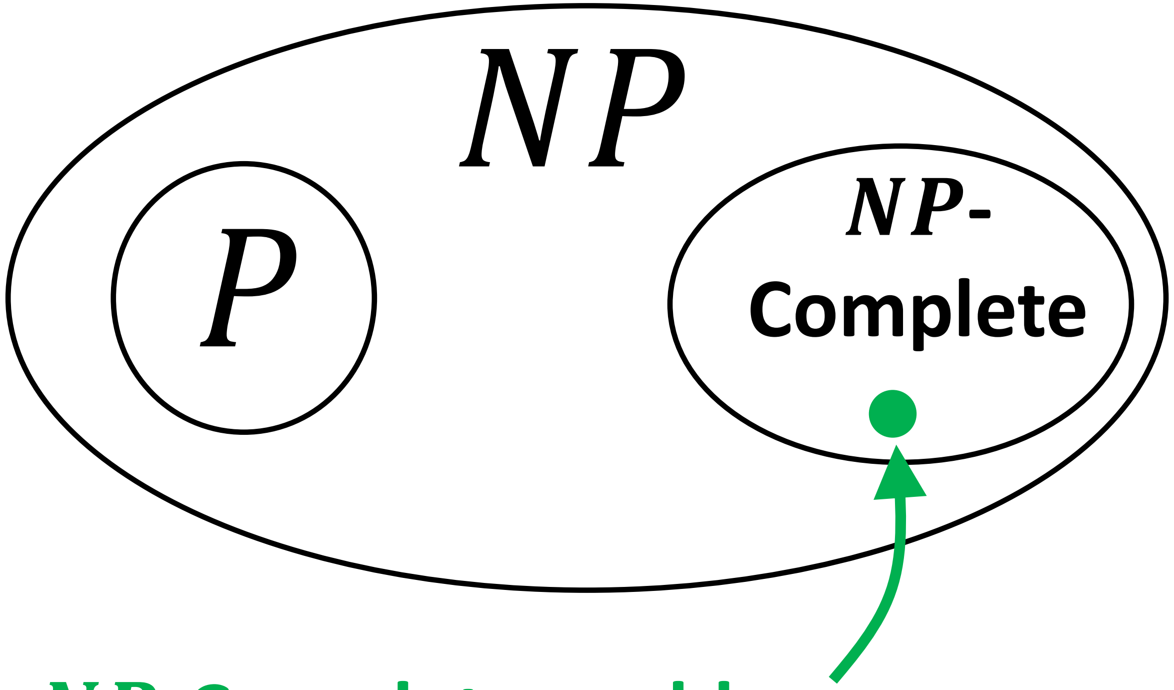
B is in NP-Complete if it satisfies two conditions:

1. $B \in NP$.
2. For every $A \in NP$, $A \leq_P B$.

B is in NP-Complete if it satisfies two conditions:

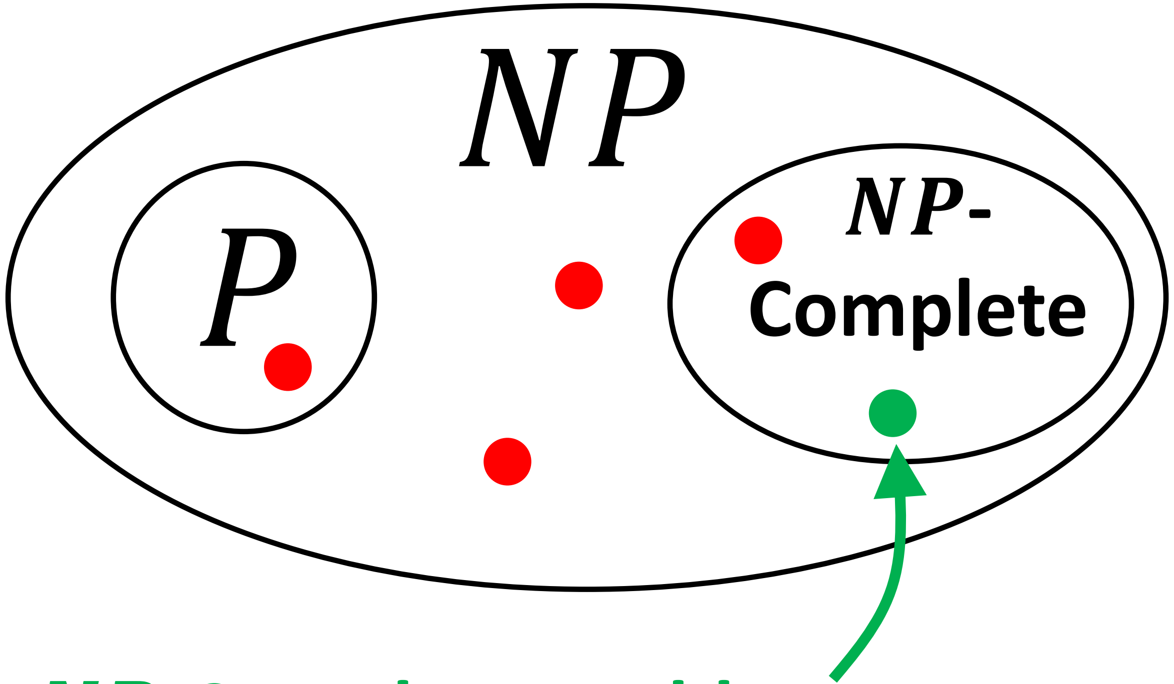
1. $B \in NP$.
2. For some $A \in NP$-C, $A \leq_P B$.

NP-Complete



A solution to an **NP-Complete problem** can be used to solve any problem in NP, with just polynomial extra time.

NP-Complete

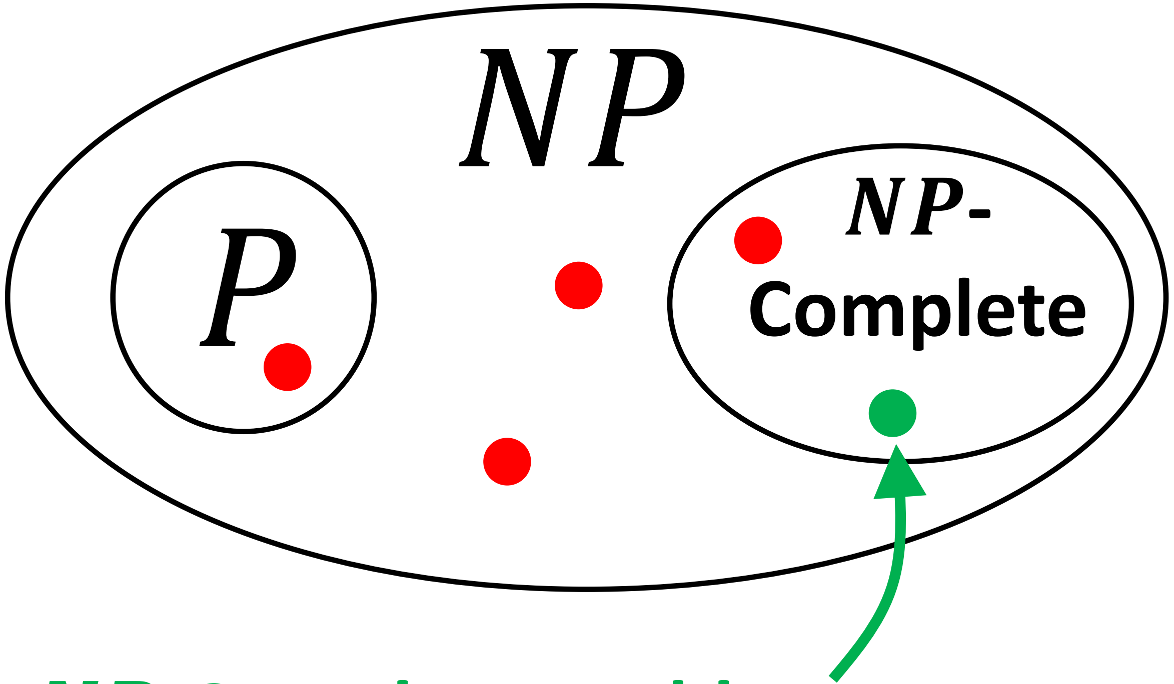


A solution to an **NP-Complete problem** can be used to solve **any problem in NP**, with just polynomial extra time.

NP-Complete

NP-Complete Problems:

- Vertex Cover
- Independent Set
- SAT
- 3-SAT

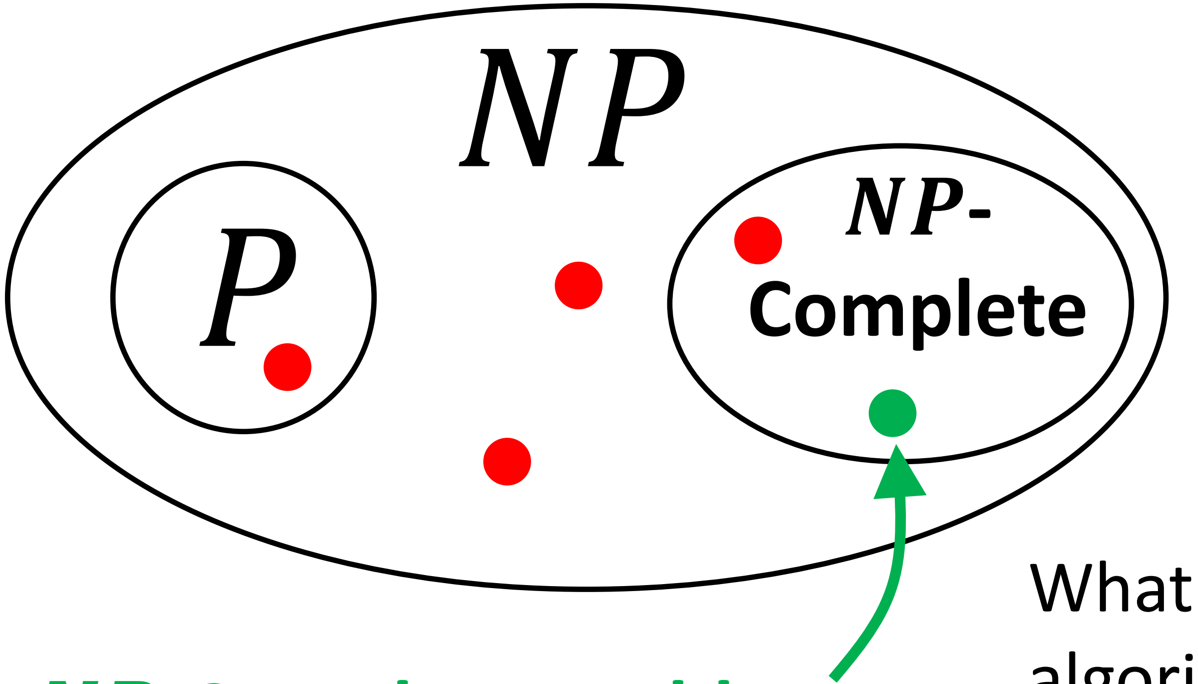


A solution to an **NP-Complete problem** can be used to solve **any problem in NP**, with just polynomial extra time.

NP-Complete

NP-Complete Problems:

- **Vertex Cover**
- Independent Set
- SAT
- 3-SAT



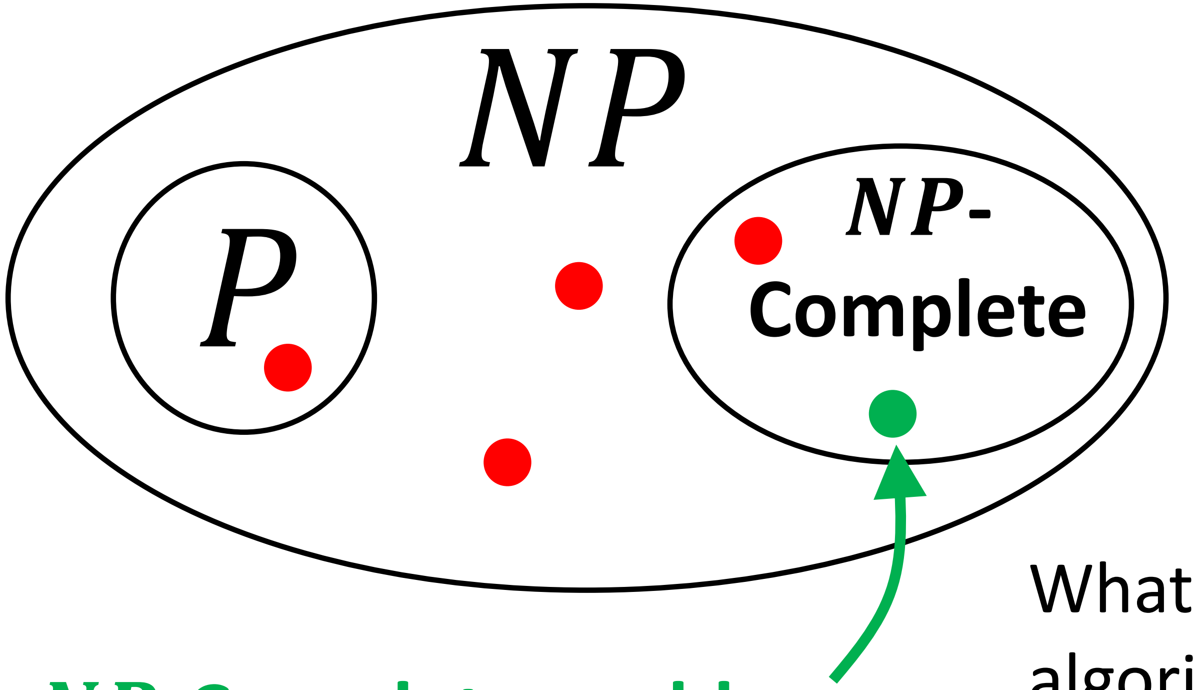
What if ∃ polynomial time algorithm for **Vertex Cover**?

A solution to an ***NP*-Complete problem** can be used to solve **any problem in *NP***, with just polynomial extra time.

NP-Complete

NP-Complete Problems:

- **Vertex Cover**
- **Independent Set**
- **SAT**
- **3-SAT**



A solution to an **NP-Complete problem** can be used to solve **any problem in NP**, with just polynomial extra time.

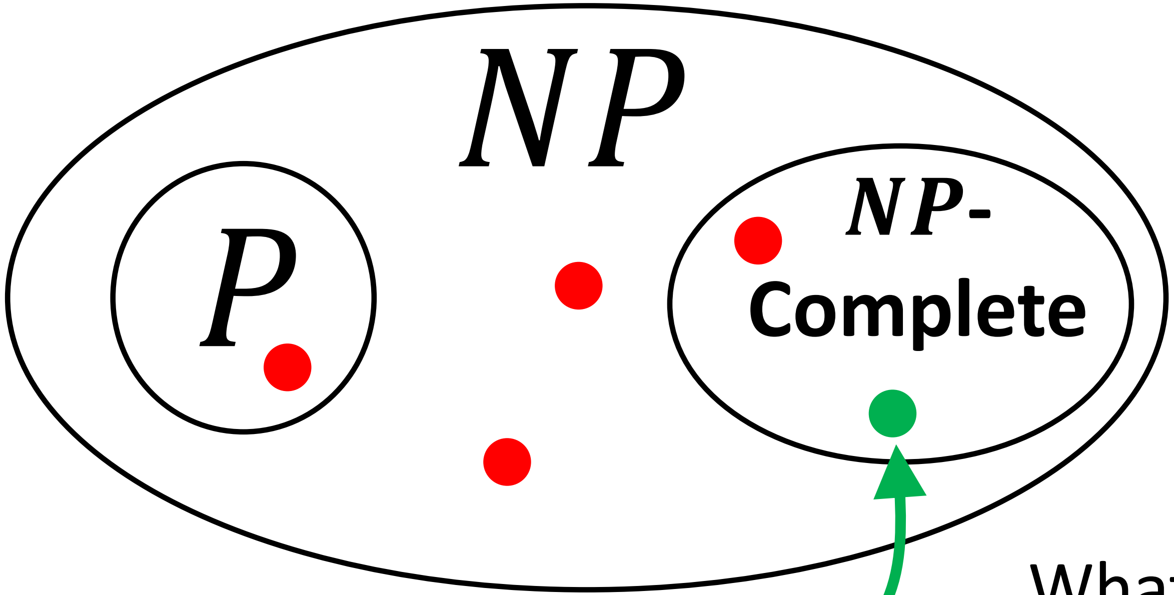
What if $\exists$ polynomial time algorithm for **Vertex Cover**?

- It could be used to solve **any problem in NP** in polynomial time.

NP-Complete

NP-Complete Problems:

- **Vertex Cover**
- **Independent Set**
- **SAT**
- **3-SAT**

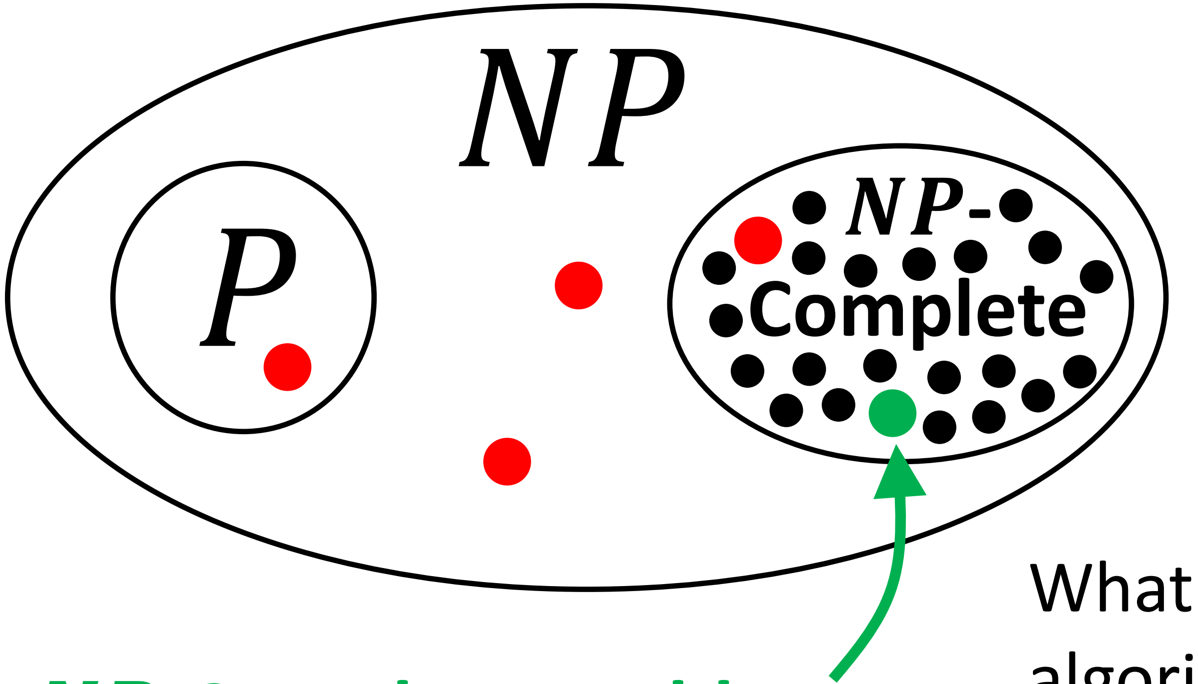


A solution to an **NP-Complete problem** can be used to solve **any problem in NP**, with just polynomial extra time.

What if $\exists$ polynomial time algorithm for **Vertex Cover**?

- It could be used to solve **any problem in NP** in polynomial time.
- $P = NP$.

NP-Complete



NP-Complete Problems:

- Vertex Cover
- Independent Set
- SAT
- 3-SAT

A solution to an **NP-Complete problem** can be used to solve **any problem in NP**, with just polynomial extra time.

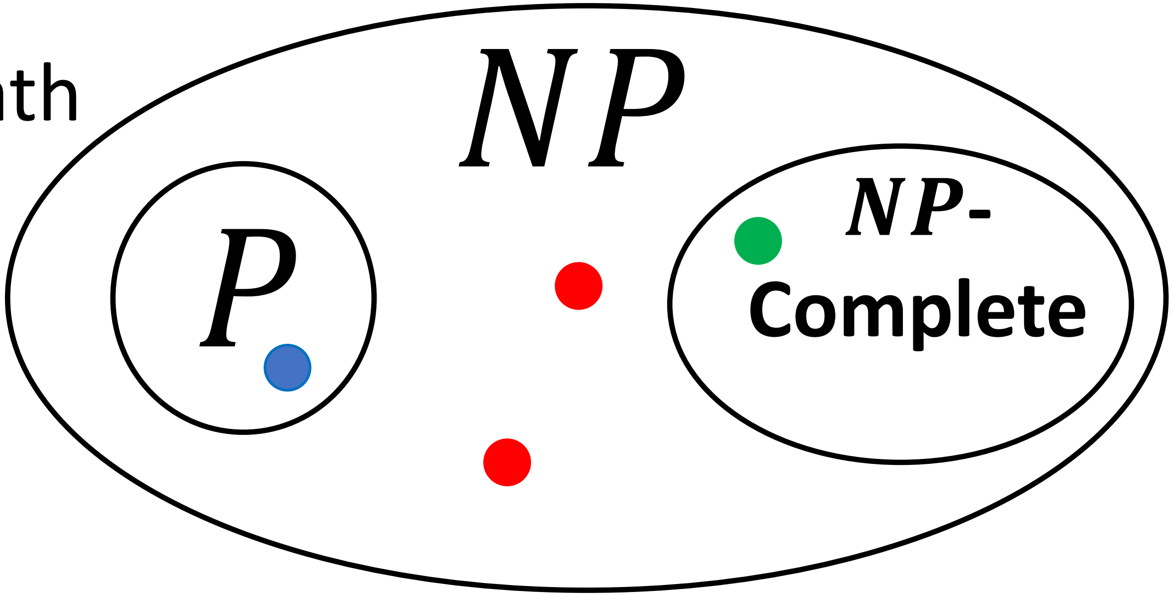
What if $\exists$ polynomial time algorithm for **Vertex Cover**?

- It could be used to solve **any problem in NP** in polynomial time.
- $P = NP$.

NP-Complete

P Problems:

- Shortest Path
- Searching
- Sorting



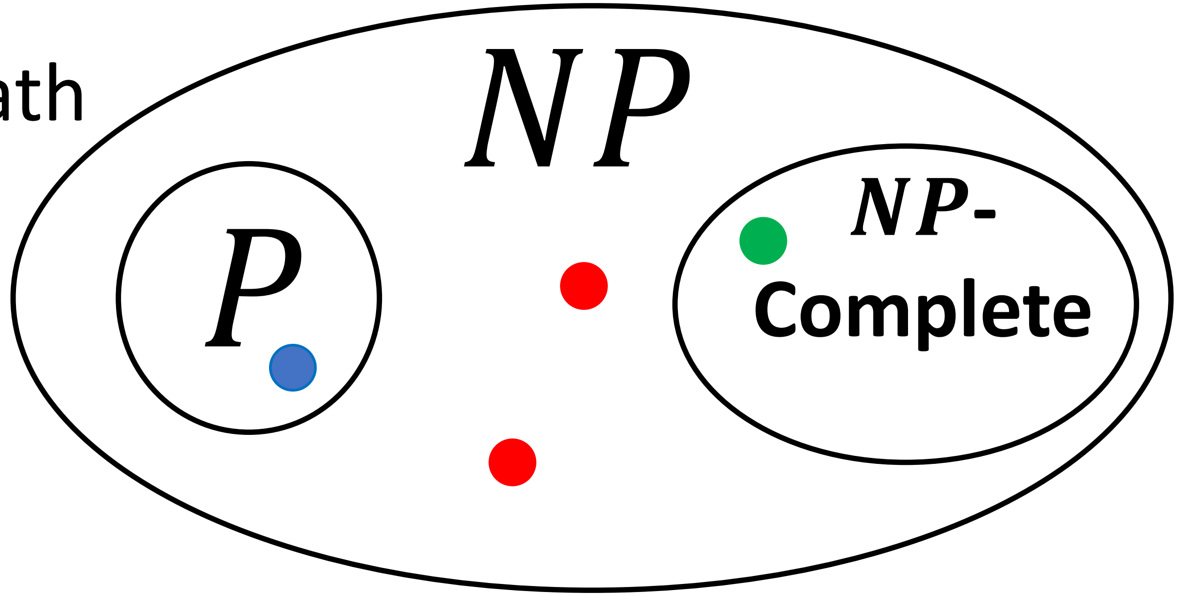
NP-Complete Problems:

- Vertex Cover
- Independent Set
- SAT
- 3-SAT

Are there **problems** in NP, but not P or NP-Complete?

NP-Complete

P Problems:
- Shortest Path
- Searching
- Sorting



NP-Complete Problems:
- Vertex Cover
- Independent Set
- SAT
- 3-SAT

Are there **problems** in NP, but not P or NP-Complete?
- We don't know. If so, $P \neq NP$.
- Suspected problems in NP but not P or NP-Complete:
  - Graph Isomorphism.
  - Integer Factorization.

NP-Complete

How to show something (B) is in NP-Complete?

B is in NP-Complete if it satisfies two conditions:

1. $B \in NP$.
2. For some $A \in NP$-C, $A \leq_P B$.