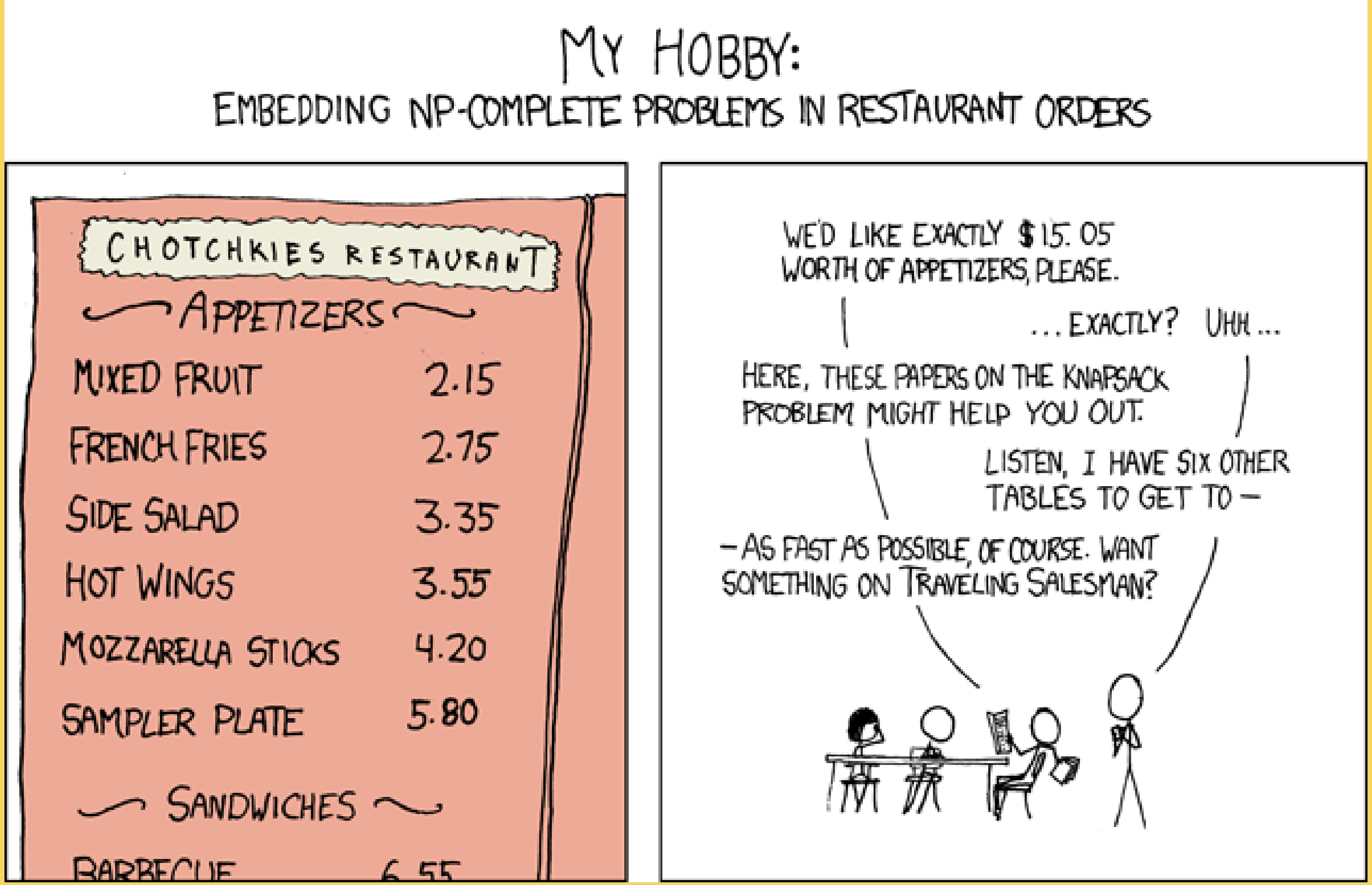
PS10 due this Friday, Apr 25.



https://xkcd.com/287/

Class 25:
Class NP
NP-Complete

University of Virginia
cs3120: DMT2
Wei-Kai Lin

Recap: Complexity Class NP

$F(x): \{ \quad \}^* \to \{0,1\}$

Informal def: decisional problem F is in NP if: whenever a problem instance $x \in F$ then this can be proved by providing a **witness** that is **polynomial-time verifiable**.

problem instance

Recap: Complexity Class NP

Formal Definition of NP:

Definition 15.1 (NP)

We say that $F : \{0,1\}^* \rightarrow \{0,1\}$ is in $\mathbf{NP}$ if there exists some integer $a > 0$ and $V : \{0,1\}^* \rightarrow \{0,1\}$ such that $V \in \mathbf{P}$ and for every $x \in \{0,1\}^n$,

$$F(x) = 1 \Leftrightarrow \exists_{w \in \{0,1\}^{n^a}} \text{ s.t. } V(xw) = 1 \;. \quad (15.1)$$

witness.

The Class P

Functions that can be computed in polynomial time by a standard Turing Machine.

$$\bigcup_{c \in \mathbb{N}} TIME_{TM}(n^c)$$

The Class NP

Functions that can be **verified** in polynomial time by a standard Turing Machine.

Correctness of a **1** output can be *verified* in polynomial time given a witness.

A function $F: \{0,1\}^* \to \{0,1\}$ is in **NP** if there exists some $a \in \mathbb{N}^+$ and $V: \{0,1\}^* \to \{0,1\}$ such that $V \in \mathbf{P}$ and $\forall x \in \{0,1\}^n, F(x) = 1 \leftrightarrow \exists w \in \{0,1\}^{n^a}$ such that $V(x, w) = 1$.

Example: 3SAT $\in$ **NP**

3SAT

Input: A Boolean formula in 3CNF form.
Output: If there is an assignment of values to variables that makes the formula to True, 1. Otherwise, 0.

A function $F: \{0,1\}^* \to \{0,1\}$ is in **NP** if there exists some $a \in \mathbb{N}^+$ and $V: \{0,1\}^* \to \{0,1\}$ such that $V \in \mathbf{P}$ and $\forall x \in \{0,1\}^n, F(x) = 1 \leftrightarrow \exists w \in \{0,1\}^{n^a}$ such that $V(x,w) = 1$.

Correctness of a 1-output can be *verified* in polynomial time given a witness.

x: $(x_1 \vee x_2 \vee \bar{x}_3) \wedge (x_3 \vee x_2 \vee x_4) \wedge (x_1 \vee x_2 \vee x_3)$ 3 CNF
F: 3SAT
w: $x_1 = 1, x_2 = 1, x_3 = 1, x_4 = 1$
V(x,w): put w into x

Example: 3SAT ∈ **NP**

3SAT

Input: A Boolean formula in 3CNF form.

Output: If there is an assignment of values to variables that makes the formula to True, 1. Otherwise, 0.

A function $F: \{0,1\}^* \rightarrow \{0,1\}$ is in **NP** if there exists some $a \in \mathbb{N}^+$ and $V: \{0,1\}^* \rightarrow \{0,1\}$ such that $V \in \mathbf{P}$ and $\forall x \in \{0,1\}^n, F(x) = 1 \leftrightarrow \exists w \in \{0,1\}^{n^a}$ such that $V(x,w) = 1$.

Correctness of a 1-output can be *verified* in polynomial time given a witness.

x: $(x_1 \vee x_2 \vee \bar{x}_3) \wedge (x_3 \vee x_2 \vee x_4) \wedge (x_1 \vee x_2 \vee x_3)$

F: 3SAT

w: $x_1 = 1, x_2 = 1, x_3 = 1, x_4 = 1$ assignment to x

V(x,w): put w into x

↔ , "if and only if":

$F(x) = 0 \Rightarrow$ not exists w such that … $V(x,w)=1$

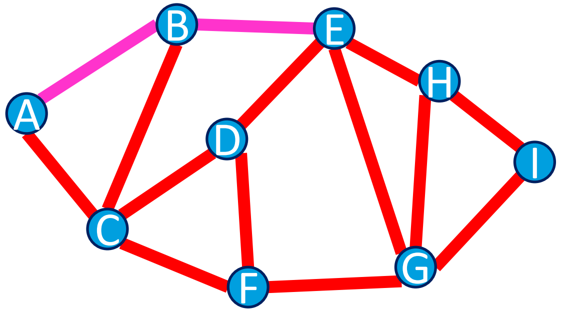
LongestPath ∈ **NP**

LongestPath (G, s, t, ℓ)

Input: A finite graph $G = (V, E)$, two vertices, $s, t \in V$, and a path length, $\ell \in \mathbb{N}$.
Output: If there is a simple path from s to t in G of length at least ℓ, 1. Otherwise, 0.

A function $F: \{0,1\}^* \rightarrow \{0,1\}$ is in **NP** if there exists some $a \in \mathbb{N}^+$ and $V: \{0,1\}^* \rightarrow \{0,1\}$ such that $V \in \mathbf{P}$ and $\forall x \in \{0,1\}^n, F(x) = 1 \leftrightarrow \exists w \in \{0,1\}^{n^a}$ such that $V(x, w) = 1$.

Correctness of a 1-output can be verified in polynomial time given a witness.



F: LongestPath
x: G, s, t, ℓ
w: a path of length ℓ
V: check the path

Example: $\mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{EXP}$

Class **P** = $\bigcup_{c\in\mathbb{N}} TIME_{TM}(n^c)$

A function $F: \{0,1\}^* \rightarrow \{0,1\}$ is in **NP** if there exists some $a \in \mathbb{N}^+$ and $V: \{0,1\}^* \rightarrow \{0,1\}$ such that $V \in \mathbf{P}$ and $\forall x \in \{0,1\}^n, F(x) = 1 \leftrightarrow \exists w \in \{0,1\}^{n^a}$ such that $V(x,w) = 1$.

Suppose $F \in \mathbf{P}$. Can we show that $F \in$ **NP** ?

Yes, proof: ∃ TM M s.t. ∀ x ∈ { }^{n}, M(x) compt F(x) in n^c steps (const c∈N)

$a =$ 0

$w =$ " "

$V(x,w) =$ M(x, w=" ") ∈ P

time of M is n^c ⇒ V ∈ TIME(n^c) ⇒ V ∈ P

Class **P** = $\cup_{c \in \mathbb{N}} TIME_{TM}(n^c)$

A function $F: \{0,1\}^* \rightarrow \{0,1\}$ is in **NP** if there exists some $a \in \mathbb{N}^+$ and $V: \{0,1\}^* \rightarrow \{0,1\}$ such that $V \in \mathbf{P}$ and $\forall x \in \{0,1\}^n, F(x) = 1 \leftrightarrow \exists w \in \{0,1\}^{n^a}$ such that $V(x,w) = 1$.

Suppose $F \in \mathbf{P}$. Can we show that $F \in$ **NP** ?

Yes, proof:

By $F \in \mathbf{P}$, there is a TM $M, c \in \mathbb{N}$ such that $M(x)$ computes $F(x)$
in n^c steps for all n-bit x.

$a = c$ 0

$w =$ “”

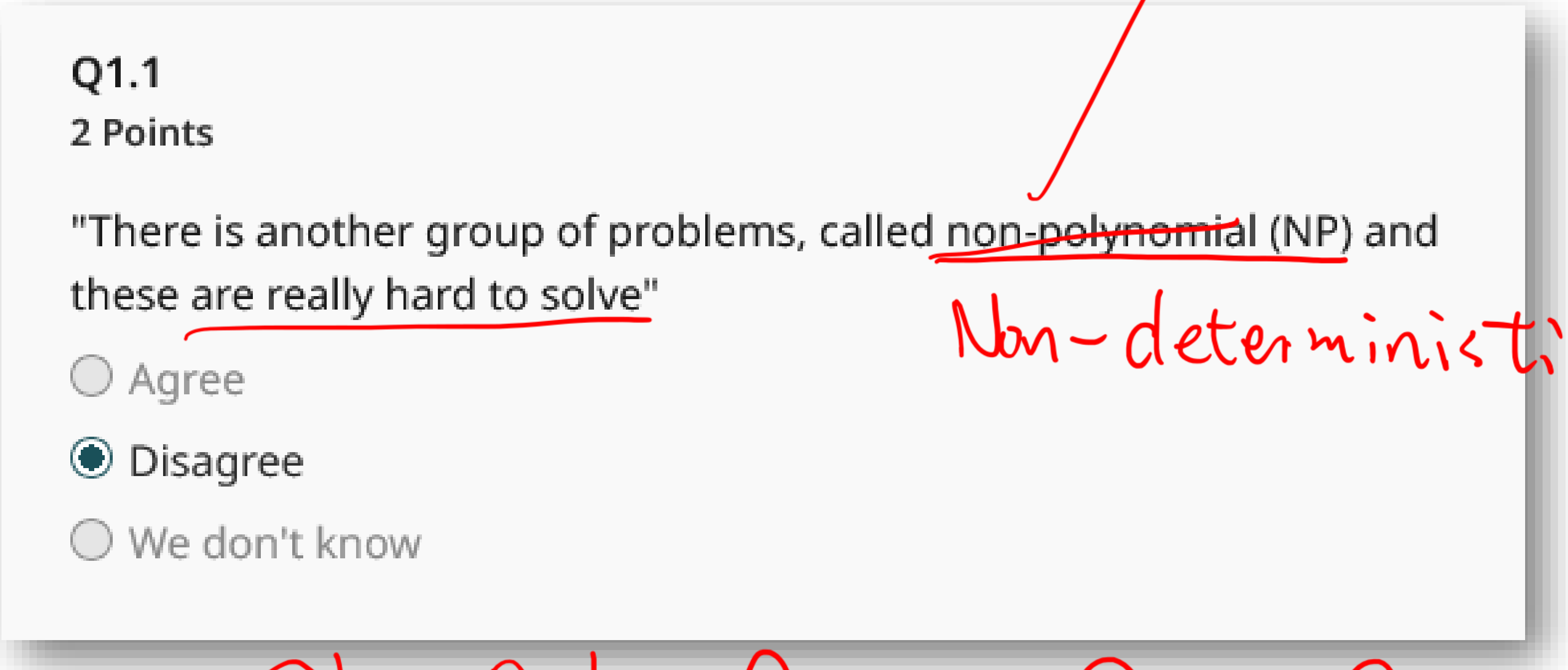
$V(x,w) = M(x)$

We have $F(x) = 1 \leftrightarrow \exists w \in \{0,1\}^{n^a}$ such that $V(x,w) = 1$.

Thus $F \in$ **NP.**

PRR12

Watch video: https://www.youtube.com/watch?v=dJUEkjxylBw


Q1.1
2 Points

"There is another group of problems, called ~~non-polynomial~~ (NP) and these are really hard to solve"

- Agree
- Disagree (selected)
- We don't know

Non-deterministic Poly

ShortPath ∈ P   SP ∈ NP

P ⊆ NP, so some problems in **NP** are not hard

A function $F: \{0,1\}^* \to \{0,1\}$ is in **NP** if there exists some $a \in \mathbb{N}^+$ and $V: \{0,1\}^* \to \{0,1\}$ such that $V \in \mathbf{P}$ and $\forall x \in \{0,1\}^n, F(x) = 1 \leftrightarrow \exists w \in \{0,1\}^{n^a}$ such that $V(x,w) = 1$.

M_V

Class **EXP** = $\bigcup_{c \in \{1,2,3,\dots\}} TIME_{TM}(2^{n^c})$

Suppose $F \in \mathbf{NP}$. We have a, V.

$M_V(x,w) = V(x,w)$

Want: $F \in$ **EXP**. TM $M(x)$ computes $F(x)$ in time 2^{n^c}.

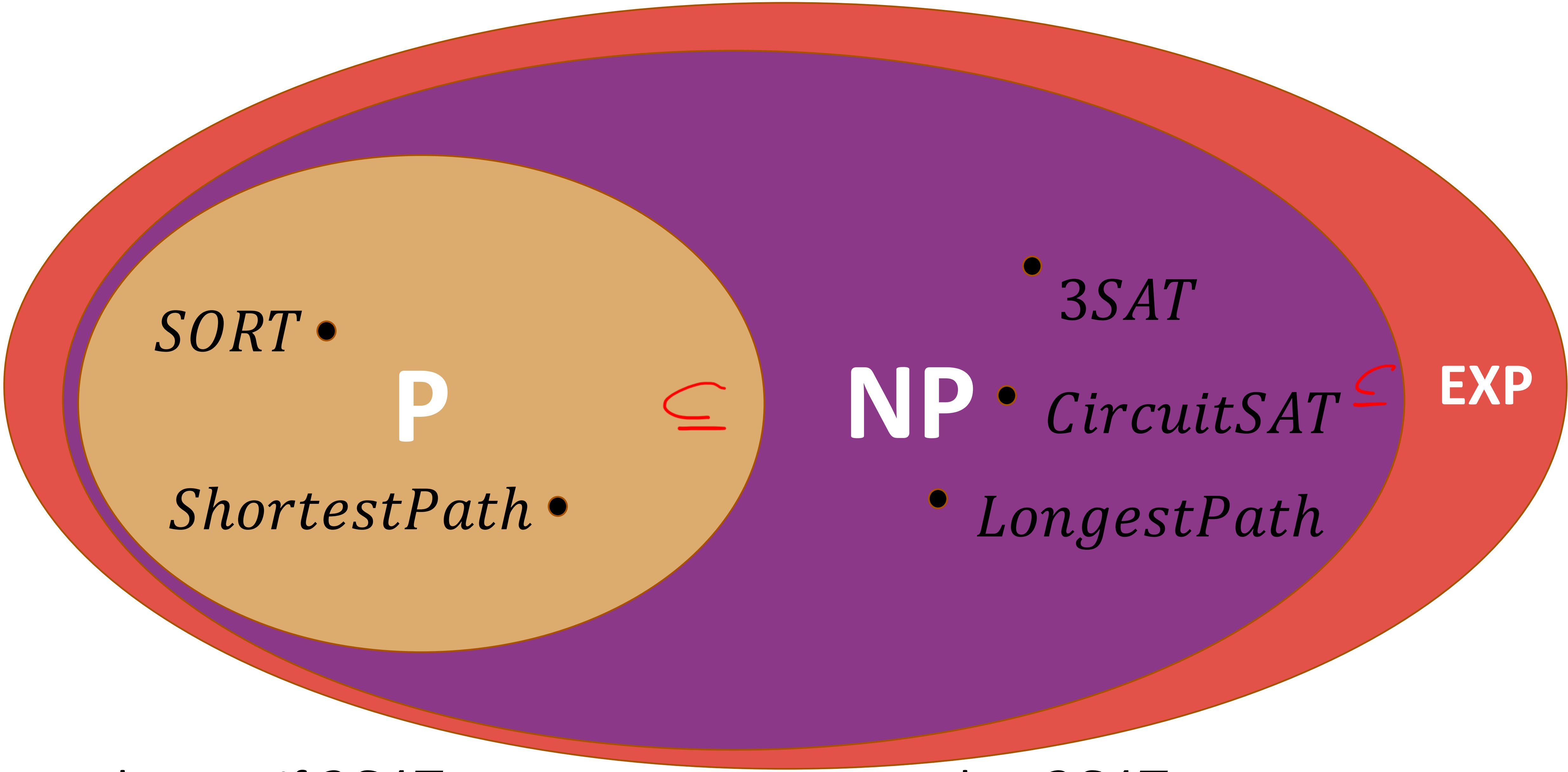
$M(x)$:

for all $w \in \{0,1\}^{n^a}$

if $M_V(x,w) = 1$

return 1

return 0



Unknown if $3SAT \in$ **P**

Known that $3SAT \in$ **NP**

More problems. Are they in NP?

Example: PRIME and COMPOSITE

A function $F: \{0,1\}^* \to \{0,1\}$ is in **NP** if there exists some $a \in \mathbb{N}^+$ and $V: \{0,1\}^* \to \{0,1\}$ such that $V \in \mathbf{P}$ and $\forall x \in \{0,1\}^n, F(x) = 1 \leftrightarrow \exists w \in \{0,1\}^{n^a}$ such that $V(x,w) = 1$.

bin string → integer

PRIME(x) =	1	if x is prime
	0	otherwise
COMPOSITE(x) =	1	if x is not prime
	0	otherwise

$x = 89$ $w = ?$

$V = ?$

Is PRIME in **P**? Hard

Is COMPOSITE in **P**?

Is PRIME in **NP**?

Is COMPOSITE in **NP**?

Eg, COMPOSITE(8633) = 1 bcs $97 \times 89 = 8633$

x w

Primality Certificate

$n = |x|$ bits

Pratt certificates [edit]

The concept of primality certificates was historically introduced by the **Pratt certificate**, conceived in 1975 by Vaughan Pratt,[1] who described its structure and proved it to have polynomial size and to be verifiable in polynomial time. It is based on the Lucas primality test, which is essentially the converse of Fermat's little theorem with an added condition to make it true:

> **Lucas' theorem**: Suppose we have an integer *a* such that:
>
> - $a^{n-1} \equiv 1 \pmod{n}$,
> - for every prime factor *q* of $n - 1$, it is not the case that $a^{(n-1)/q} \equiv 1 \pmod{n}$.
>
> Then *n* is prime.

$x, \; w = \sqrt{x}$

V: div x by $y \le w$

$2^{n/2}$

Given such an *a* (called a *witness*) and the prime factorization of $n - 1$, it's simple to verify the above conditions quickly: we only need to do a linear number of modular exponentiations, since every integer has fewer prime factors than bits, and each of these can be done by exponentiation by squaring in $O(\log n)$ multiplications (see big-O notation). Even with grade-school integer multiplication, this is only $O((\log n)^4)$ time; using the multiplication algorithm with best-known asymptotic running time, due to David Harvey and Joris van der Hoeven, we can lower this to $O((\log n)^3(\log \log n))$ time, or using soft-O notation $\tilde{O}((\log n)^3)$.

However, it is possible to trick a verifier into accepting a composite number by giving it a "prime factorization" of $n - 1$ that includes composite numbers. For example, suppose we claim that $n = 85$ is prime, supplying $a = 4$ and $n - 1 = 6 \times 14$ as the "prime factorization". Then (using $q = 6$ and $q = 14$):

To be continued ... https://en.wikipedia.org/wiki/Primality_certificate

Annals of Mathematics, **160** (2004), 781–793

PRIMES is in P

By Manindra Agrawal, Neeraj Kayal, and Nitin Saxena*

Abstract

We present an unconditional deterministic polynomial-time algorithm that determines whether an input number is prime or composite.

https://annals.math.princeton.edu/wp-content/uploads/annals-v160-n2-p12.pdf

3UNSAT $\in$? **NP**

3UNSAT

Input: A Boolean formula in 3CNF form.

Output: If there is an assignment of values to variables that makes the formula to True, **0**. Otherwise, **1**.

3UNSAT(x) = NOT(3SAT(x))

A function $F: \{0,1\}^* \rightarrow \{0,1\}$ is in **NP** if there exists some $a \in \mathbb{N}^+$ and $V: \{0,1\}^* \rightarrow \{0,1\}$ such that $V \in \mathbf{P}$ and $\forall x \in \{0,1\}^n, F(x) = 1 \leftrightarrow \exists w \in \{0,1\}^{n^a}$ such that $V(x,w) = 1$.

Correctness of a **1**-output can be *verified* in polynomial time given a witness.

3SAT $\in$ NP

Input: A Boolean formula in 3CNF form.

Output: If there is an assignment of values to variables that makes the formula to True, 1. Otherwise, 0.

NonLongestPath (G, s, t, ℓ)

Input: A finite graph $G = (V, E)$, two vertices, $s, t \in V$, and a path length, $\ell \in \mathbb{N}$.

Output: If there is a simple path from s to t in G of length at least ℓ, 0. Otherwise, 1.

LongestPath (G, s, t, ℓ)

Input: A finite graph $G = (V, E)$, two vertices, $s, t \in V$, and a path length, $\ell \in \mathbb{N}$.

Output: If there is a simple path from s to t in G of length at least ℓ, 1. Otherwise, 0.

NonLongestPath (G, s, t, ℓ) **= _NOT(LongestPath_** (G, s, t, ℓ) **_)_**

NonLongestPath $\in$? NP

ShortestPath(G, s, t, ℓ):

1 iff there is a path from s to t in G with $\leq \ell$ steps

NonLongestPath (G, s, t, ℓ)

Input: A finite graph $G = (V, E)$, two vertices, $s, t \in V$, and a path length, $\ell \in \mathbb{N}$.

Output: If there is a simple path from s to t in G of length at least ℓ, 0. Otherwise, 1.

LongestPath (G, s, t, ℓ)

Input: A finite graph $G = (V, E)$, two vertices, $s, t \in V$, and a path length, $\ell \in \mathbb{N}$.

Output: If there is a simple path from s to t in G of length at least ℓ, 1. Otherwise, 0.

NonLongestPath (G, s, t, ℓ) ***= NOT(LongestPath*** (G, s, t, ℓ) ***)***

NonLongestPath(G, A, E, 10) = 1

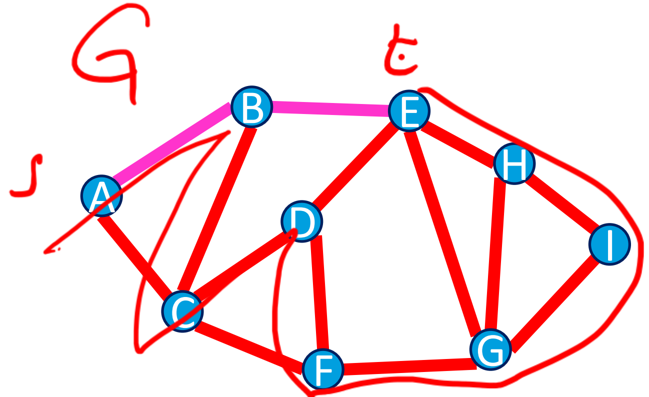
ShortestPath(G, A, E, 10) = 1

NonLongestPath(G, A, E, 5) = 0

ShortestPath(G, A, E, 5) = 1

ShortestPath(G, s, t, ℓ):

1 iff there is a path from s to t in G with $\leq \ell$ steps



Class **co-NP**

A function $F: \{0,1\}^* \rightarrow \{0,1\}$ is in **NP** if there exists some $a \in \mathbb{N}^+$ and $V: \{0,1\}^* \rightarrow \{0,1\}$ such that $V \in \mathbf{P}$ and $\forall x \in \{0,1\}^n, F(x) = 1 \leftrightarrow \exists w \in \{0,1\}^{n^a}$ such that $V(x,w) = 1$.

Class **co-NP** = $\{F: F$ is a Boolean function and $NOT\big(F(x)\big) \in$ **NP** $\}$

3UNSAT
Input: A Boolean formula in 3CNF form.
Output: If there is an assignment of values to variables that makes the formula to True, $\mathbf{0}$. Otherwise, $\mathbf{1}$.

3SAT
Input: A Boolean formula in 3CNF form.
Output: If there is an assignment of values to variables that makes the formula to True, 1. Otherwise, 0.

Class **co-NP**

A function $F: \{0,1\}^* \to \{0,1\}$ is in **NP** if there exists some $a \in \mathbb{N}^+$and $V: \{0,1\}^* \to \{0,1\}$ such that $V \in \mathbf{P}$ and $\forall x \in \{0,1\}^n, F(x) = 1 \leftrightarrow \exists w \in \{0,1\}^{n^a}$ such that $V(x,w) = 1$.

Class **co-NP** = $\{F: F$ is a Boolean function and $NOT\big(F(x)\big) \in$ **NP** $\}$

P $\subseteq$ **co-NP**

We do not know the inclusion between **NP** and **co-NP**

Open Problem: $\mathbf{P} = \mathbf{NP}$?

Millennium Problem ($1 million prize)

Clay Mathematics Institute

About Programs & Awards People The Millennium Prize Problems Online resources Events News

Home — Millennium Problems — P vs NP

Unsolved

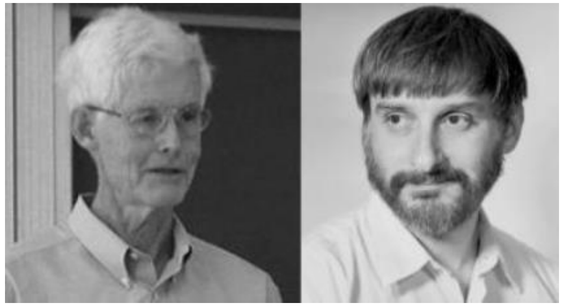
P vs NP

If it is easy to check that a solution to a problem is correct, is it also easy to solve the problem? This is the essence of the P vs NP question. Typical of the NP problems is that of the Hamiltonian Path Problem: given N cities to visit, how can one do this without visiting a city twice? If you give me a solution, I can easily check that it is correct. But I cannot so easily find a solution.

https://www.claymath.org/millennium/p-vs-np/

Yet another problem in NP

- How to accommodate 400 students in a dorm?
- Space is limited: only 100 places
- Some pairs of incompatible students such that no pair appear in final choice



Stephen Cook and Leonid Levin

https://www.claymath.org/millennium/p-vs-np/

Open Problem: $\mathbf{P} = \mathbf{NP}$?

Is it harder to find a solution to a problem or to check if a provided solution is correct?

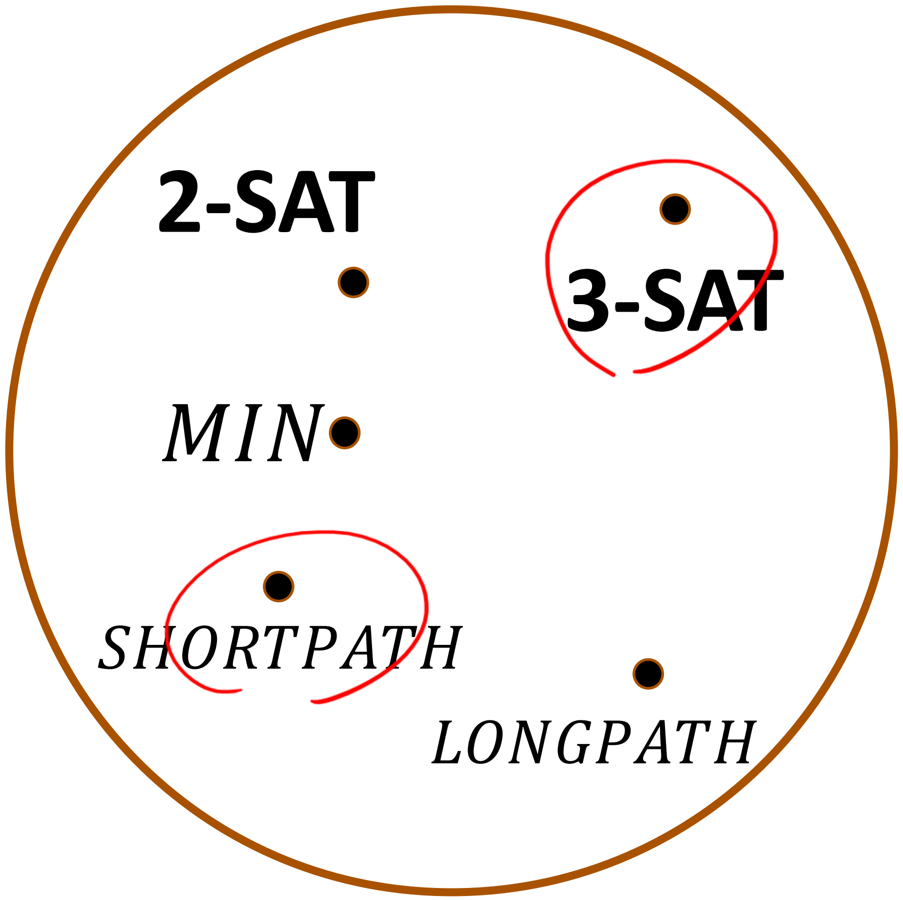
Yes: $\mathbf{P} \subsetneq \mathbf{NP}$

There are some problems where it is hard to find a solution, but if given an answer it is easy to check it.

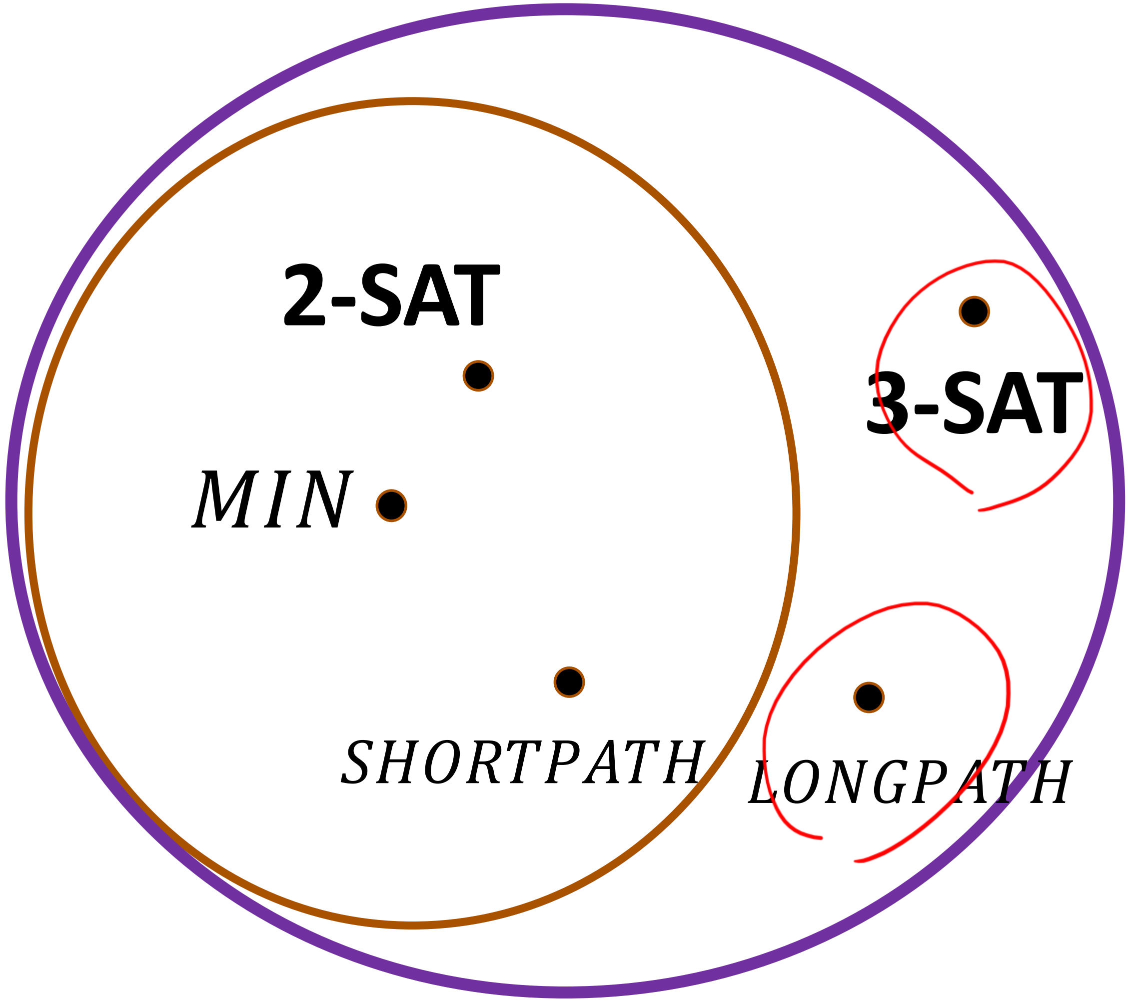
No: $\mathbf{P} = \mathbf{NP}$

If it is easy to check if a solution to a problem is correct, it is also easy to find a solution to that problem.

The answer to this question is unknown! This is the most famous open problem in mathematics and computer science.

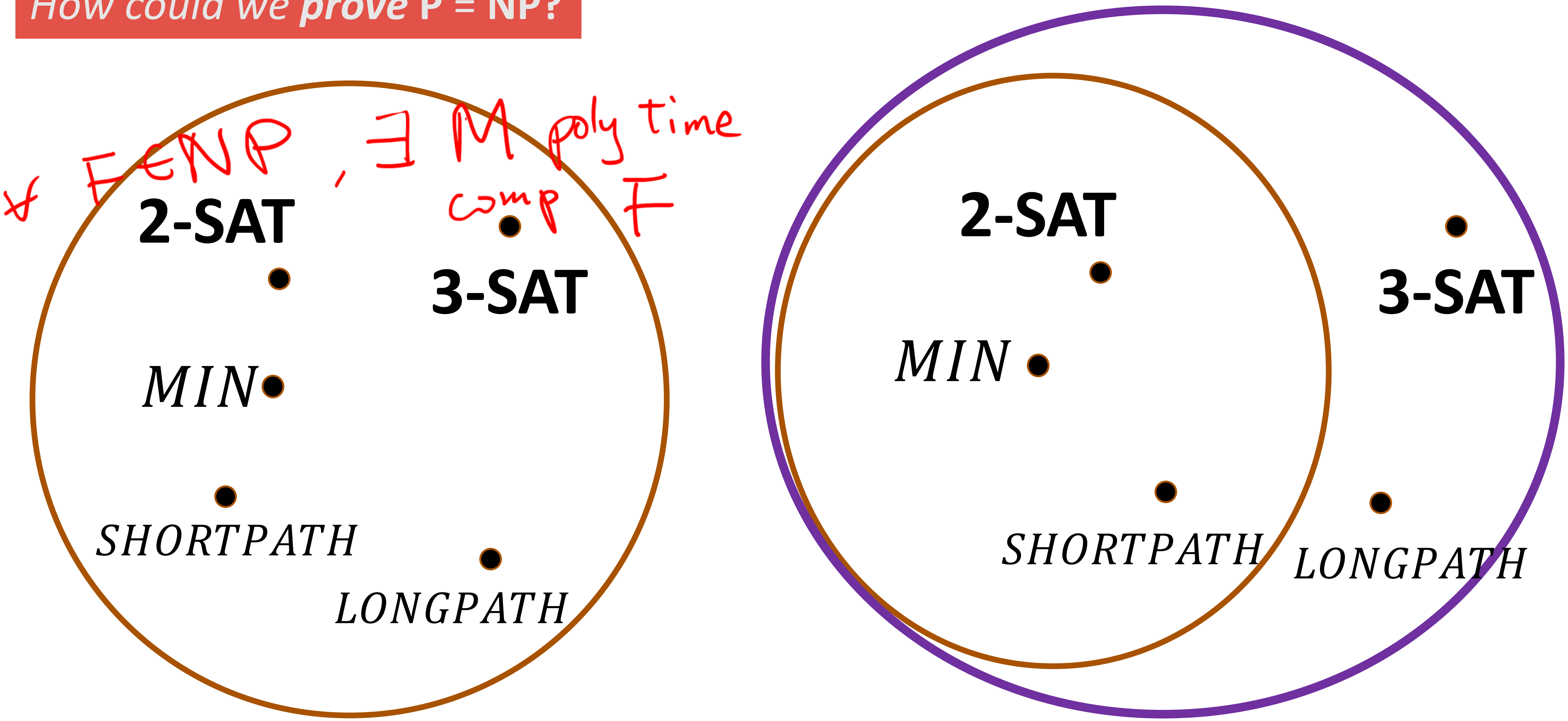


If **P = NP**



If **P** $\subsetneq$ **NP**

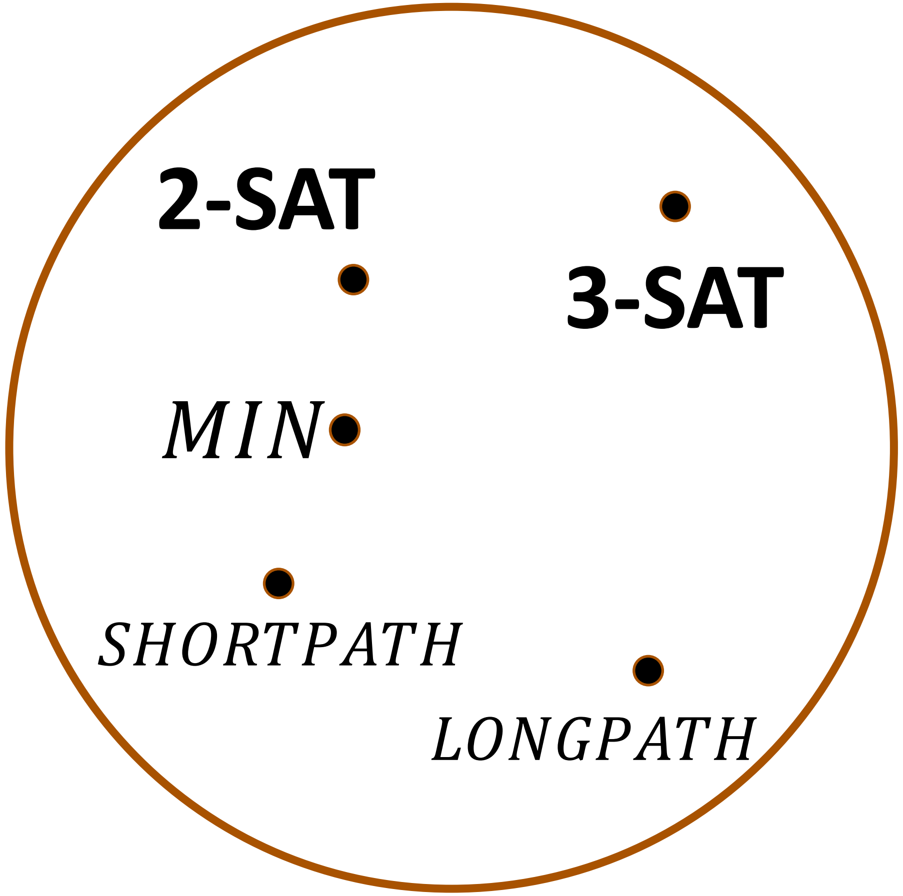
*How could we **prove** P = NP?*



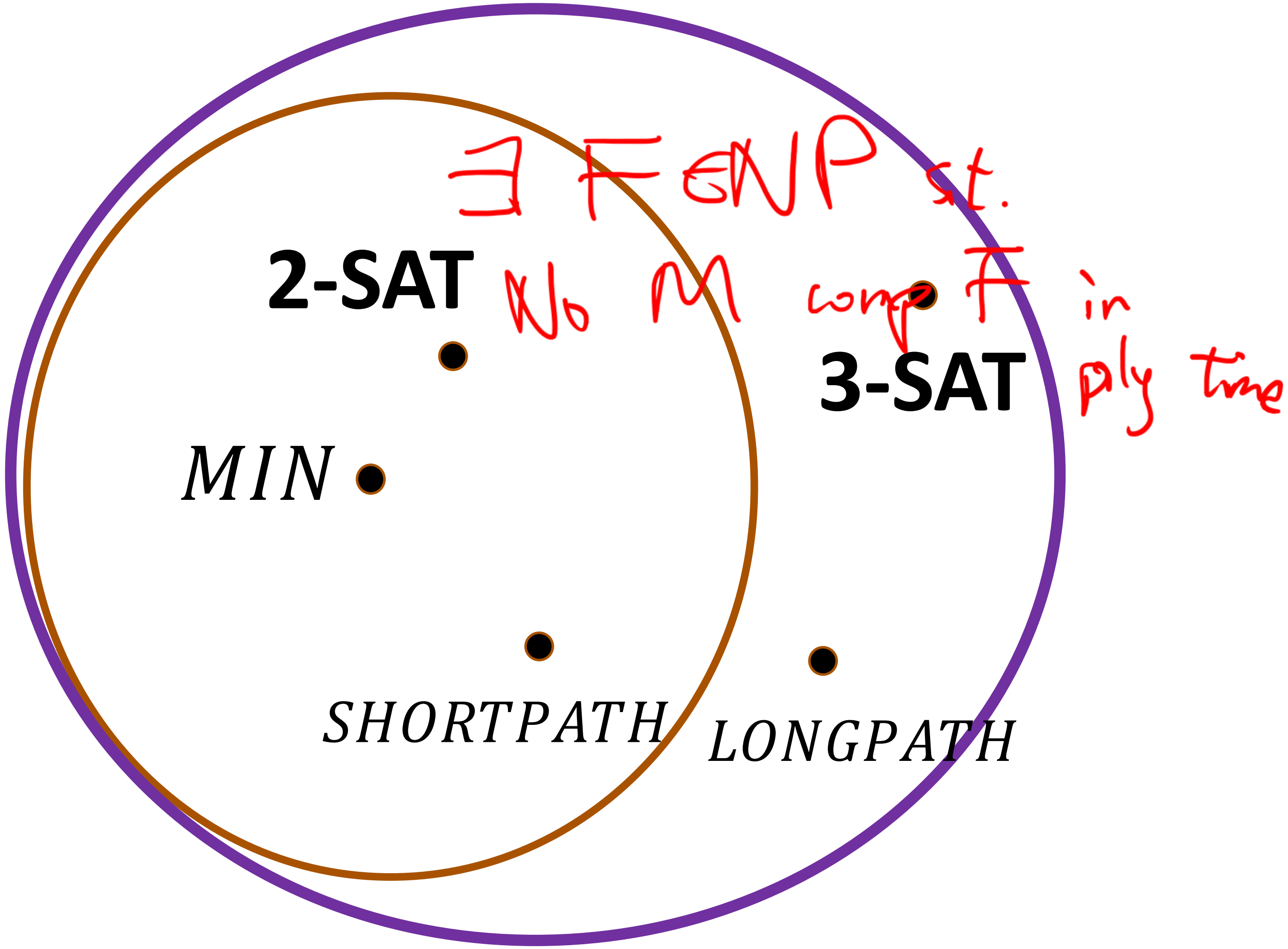
If P = NP

If P $\subsetneq$ NP

How could we ***prove*** **P $\subsetneq$ NP?**



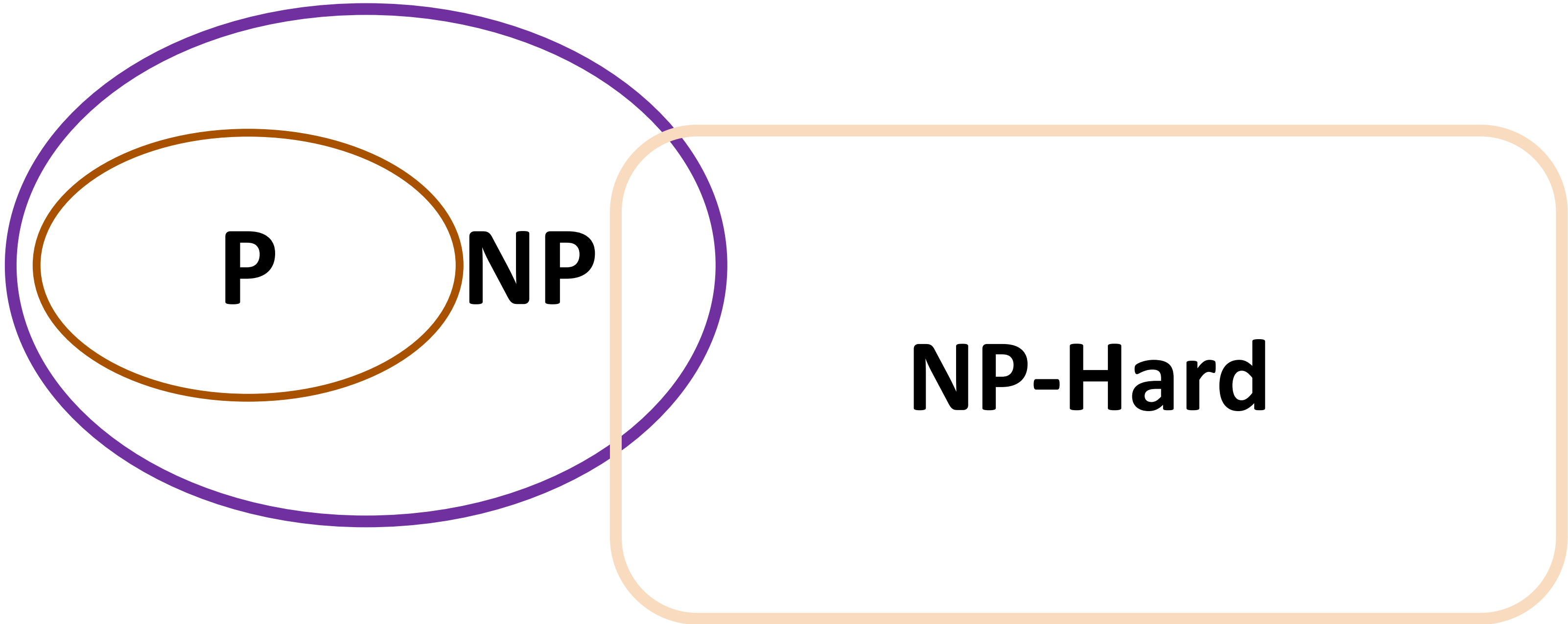
If P = NP



If P $\subsetneq$ NP

Complexity Class: NP-Hard

Definition: A Boolean function G is **NP-Hard** if every $F \in \mathbf{NP}$ can be reduced to G: $F \leq_P G$.



Cook-Levin Theorem

Definition: A function G is **NP-Hard** if every $F \in \mathbf{NP}$ can be reduced to G: $F \leq_P G$.

Cook-Levin Theorem (Theorem 15.6 in the TCS Book):

For every $F \in \mathbf{NP}$, $F \leq_p$ **3SAT.**

Cook-Levin Theorem

Definition: A function G is **NP-Hard** if every $F \in \mathbf{NP}$ can be reduced to G: $F \leq_P G$.

Cook-Levin Theorem (Theorem 15.6 in the TCS Book):

For every $F \in \mathbf{NP}$, $F \leq_p$ **3SAT.**

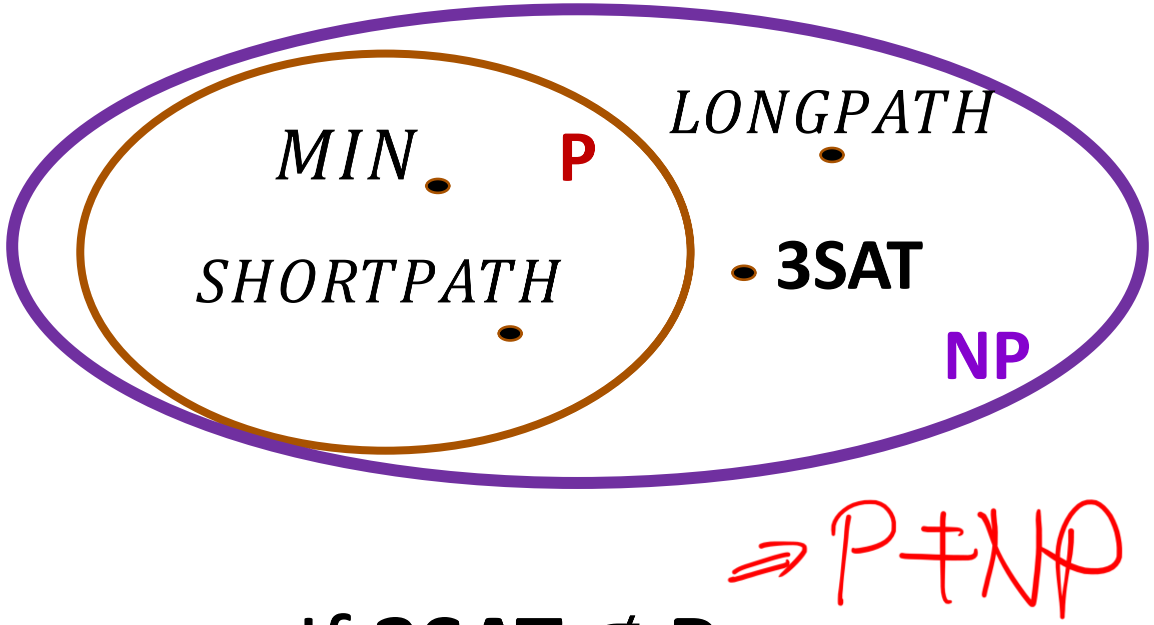
Equivalently: **3SAT** is (in) **NP-Hard**

Making Progress on $\mathbf{P} \subsetneq \mathbf{NP}$

Cook-Levin Theorem (Theorem 15.6 in the TCS Book):

For every $F \in \mathbf{NP}$, $F \leq_p$ **3SAT.**

If **3SAT** $\in$ **P**



If **3SAT** $\notin$ **P**

NP-Complete

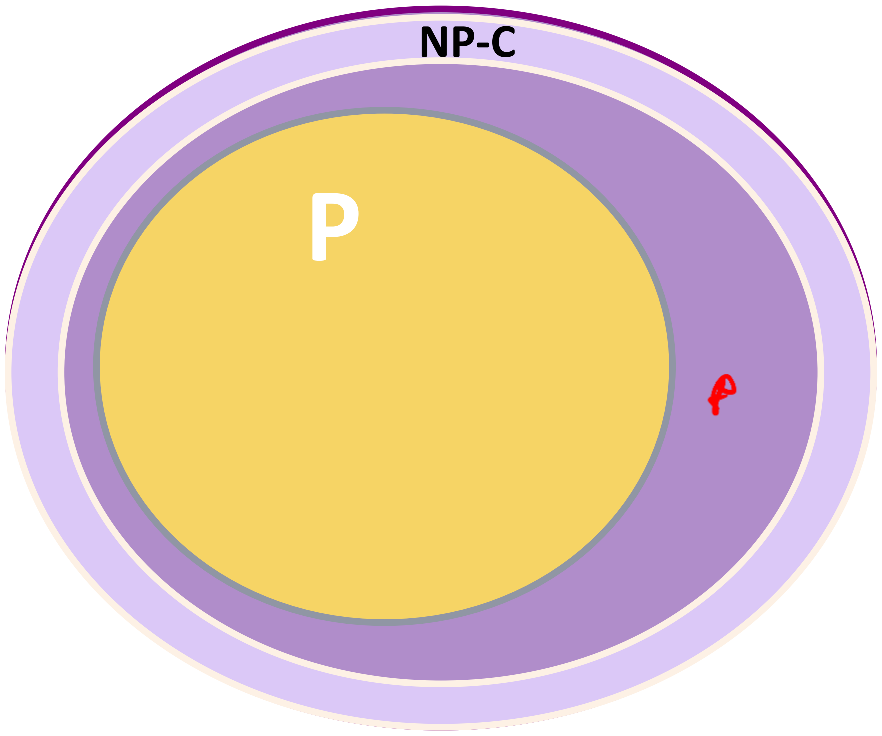
Definition: A function G is **NP-Hard** if every $F \in \mathbf{NP}$ can be reduced to G: $F \leq_P G$.

Definition: A function G is **NP-Complete** if $G \in \mathbf{NP}$ and G is **NP-Hard**.

Cook: **3SAT** is **NP-Hard**

⇒ **3SAT** is **NP-Complete** by **3SAT** ∈ **NP**

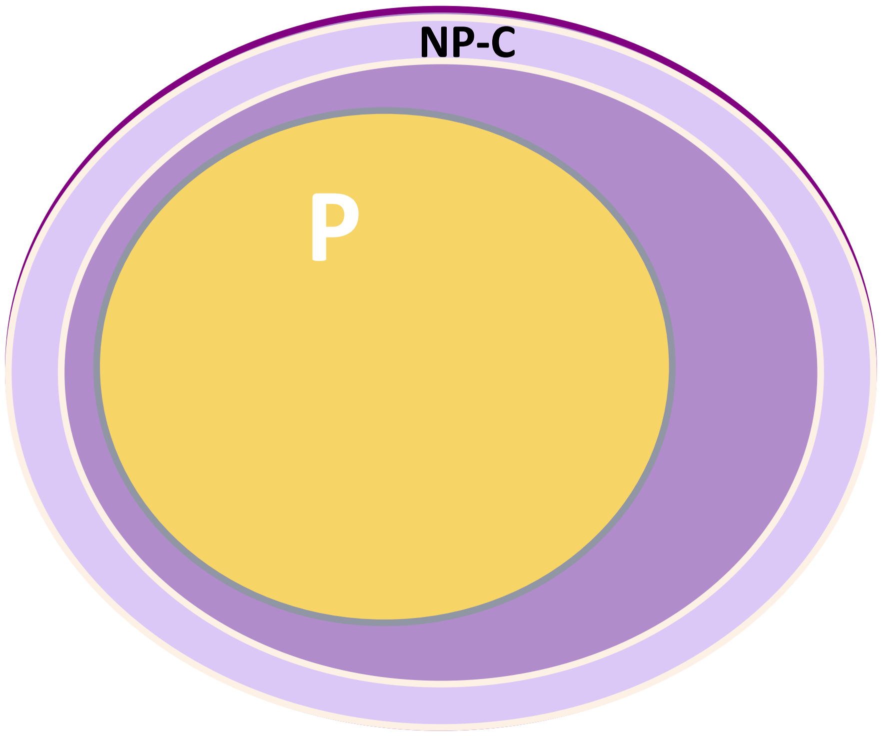
If P ⊊ NP



P ⊊ NP,

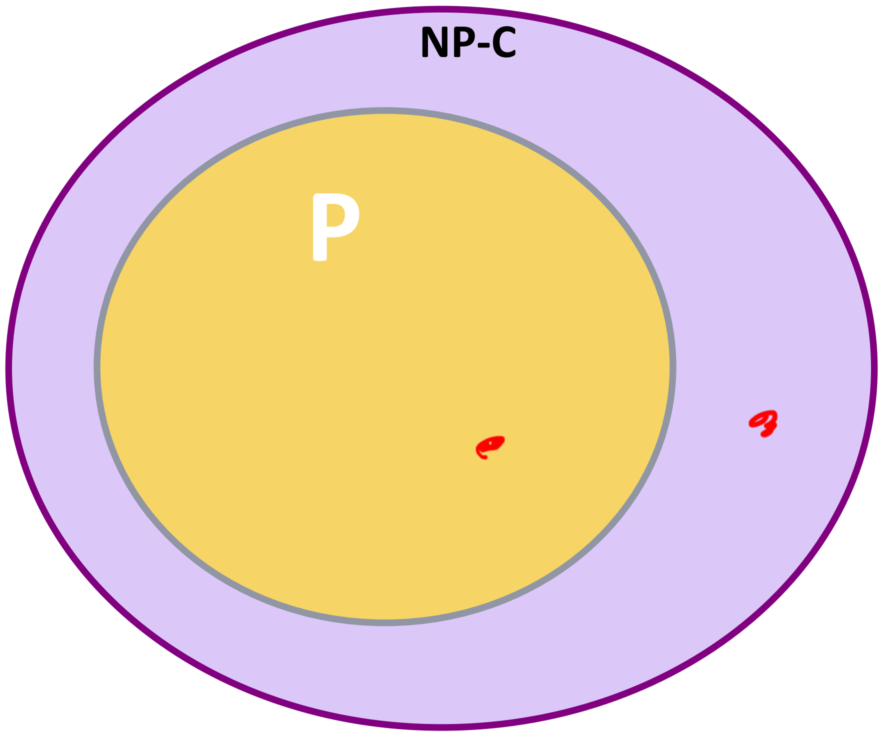
NP-C ∪ P ⊊ NP

If $\mathbf{P} \subsetneq \mathbf{NP}$



$\mathbf{P} \subsetneq \mathbf{NP}$,

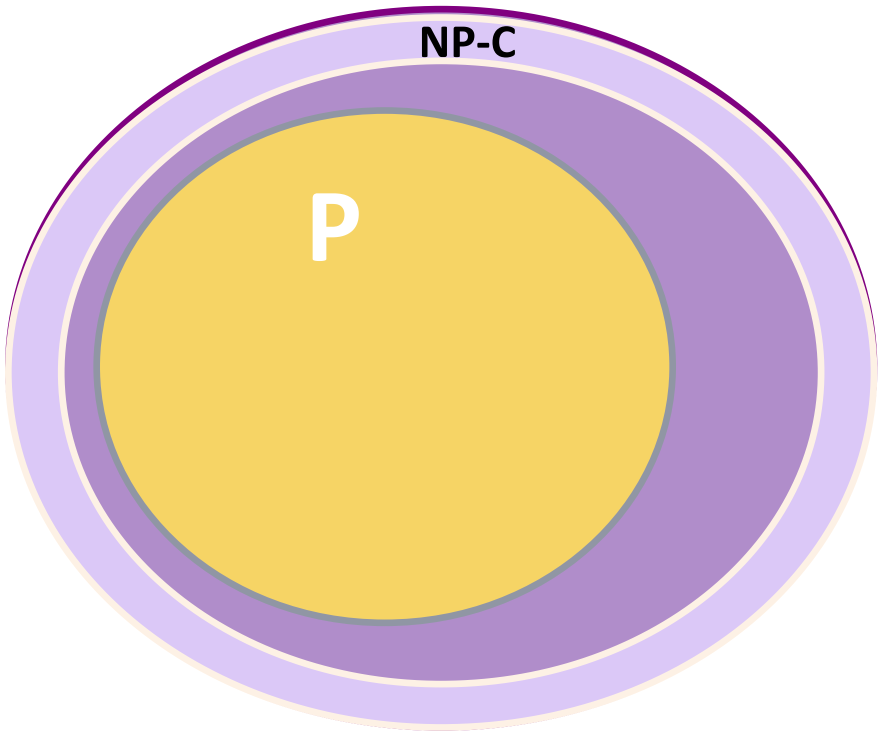
$\mathbf{NP\text{-}C} \cup \mathbf{P} \subsetneq \mathbf{NP}$



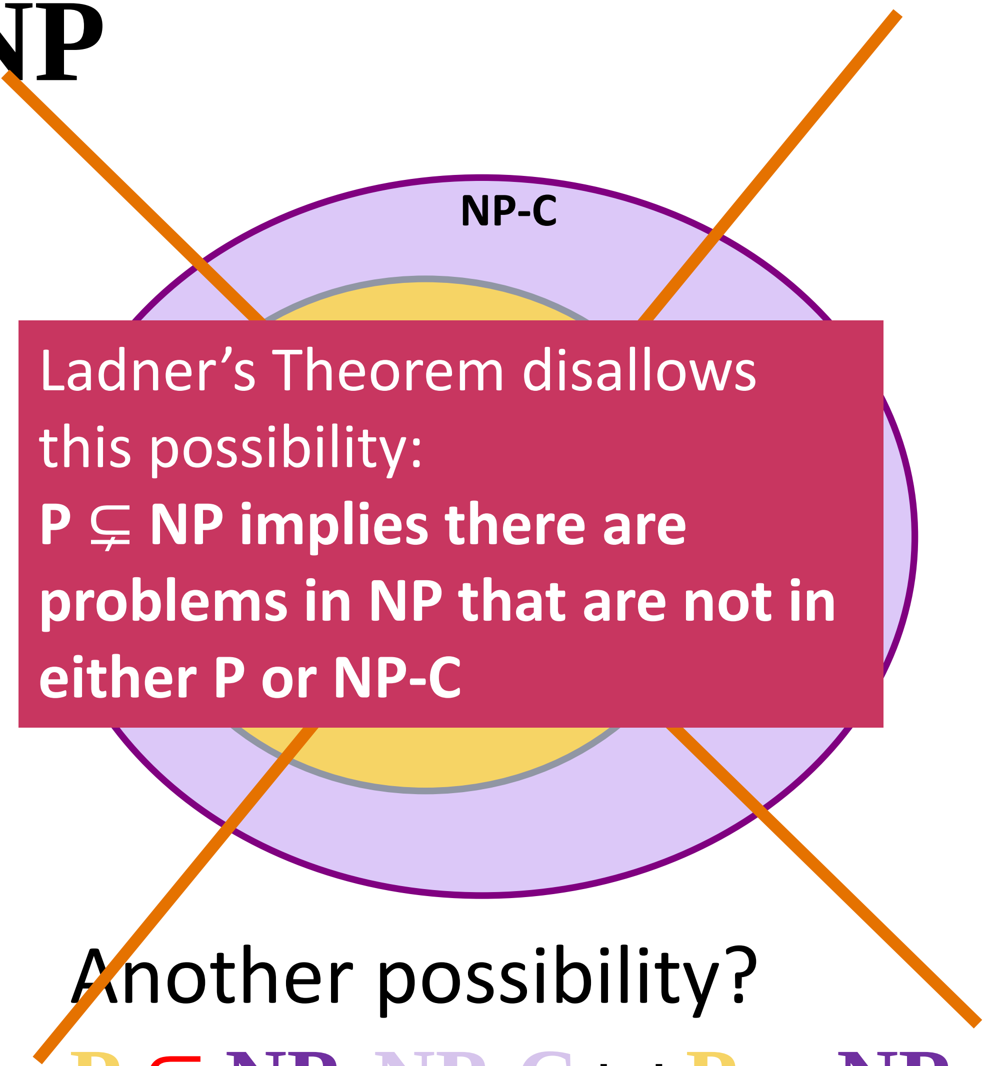
Another possibility?

$\mathbf{P} \subsetneq \mathbf{NP}$, $\mathbf{NP\text{-}C} \cup \mathbf{P} = \mathbf{NP}$

If $P \subsetneq NP$



$P \subsetneq NP$,
$NP\text{-}C \cup P \subsetneq NP$



Ladner's Theorem disallows this possibility:
$P \subsetneq NP$ implies there are problems in NP that are not in either P or NP-C

Another possibility?
$P \subsetneq NP$, $NP\text{-}C \cup P = NP$

PRR12

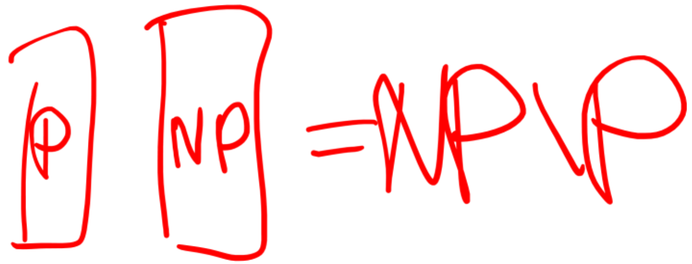
Watch video: https://www.youtube.com/watch?v=dJUEkjxylBw

Q1.5

2 Points

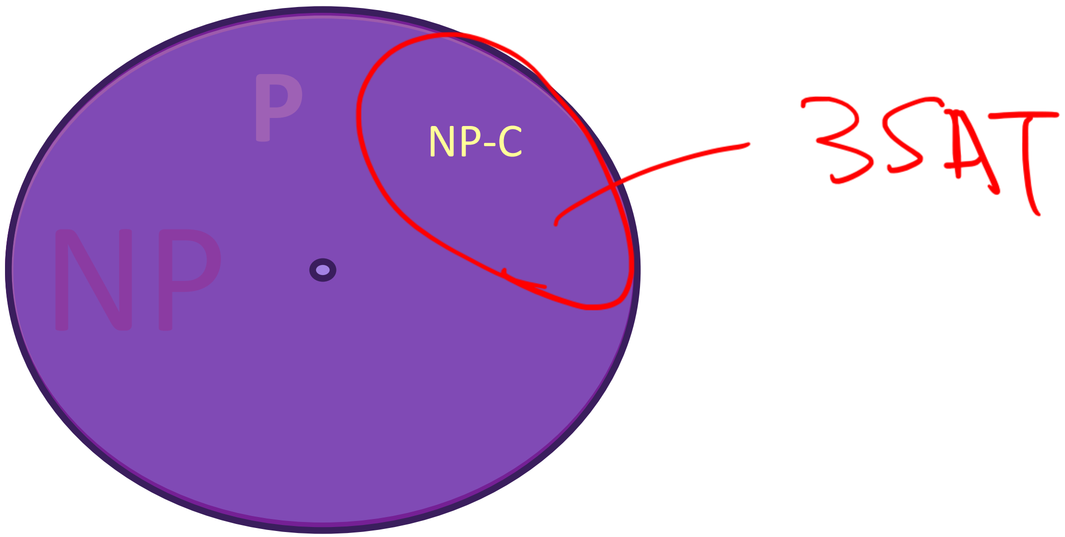
"Jeff Westbrook is saying: P and NP are fundamentally different, and they are in separate folders, and you have to keep them in separate folders"

- ○ Agree (two separate folders)
- ○ Disagree (one folder)
- ○ Disagree (three folders)
- ◉ We don't know / other



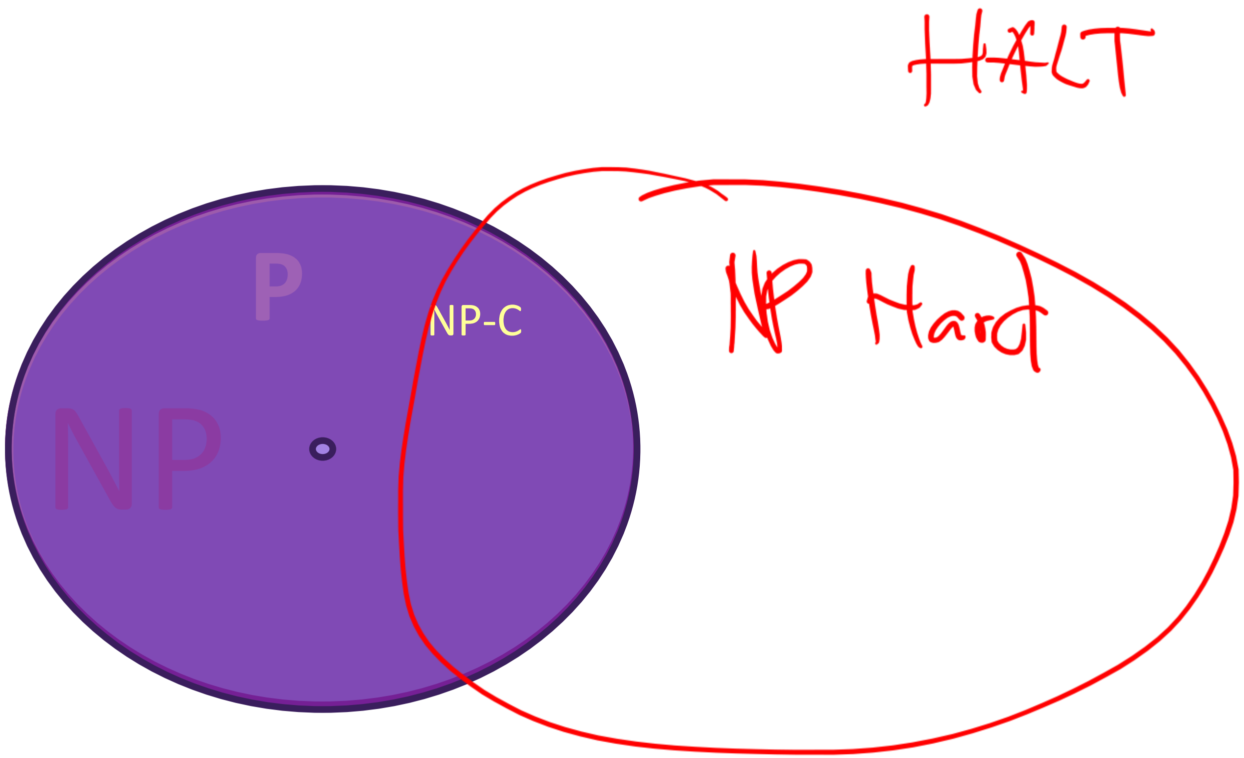
P, NP-Complete, and there are "neither" problems by Ladner's Thm

If P = NP



Option 2: **P** = **NP** ≈ **NP**-Complete

If P = NP



Option 2: **P** = **NP** ≈ **NP**-Complete

NP-Hard = All Non-Constant Functions: $\{0,1\}^* \rightarrow \{0,1\}$

Wrong! There are many functions not in **P**

Alternate Definition:
Nondeterministic Turing Machines

Standard TM Definition

A *Turing Machine,* is defined by (Σ, k, δ):

$\boldsymbol{k} \in \mathbb{N}$: a finite number of states

Σ : alphabet − finite set of symbols

$$\Sigma \supseteq \{0, 1, \rhd, \emptyset\}$$

$\boldsymbol{\delta}$: transition function

$$\boldsymbol{\delta}: [k] \times \Sigma \rightarrow [k] \times \Sigma \times \{\mathbf{L}, \mathbf{R}, \mathbf{S}, \mathbf{H}\}$$

How should we define a ***Nondeterministic Turing Machine****?*

Nondeterminstic TM

A **Nondeterministic** *Turing Machine,* is defined by (Σ, k, δ):

$\boldsymbol{k} \in \mathbb{N}$: a finite number of states

Σ : alphabet $-$ finite set of symbols

$$\Sigma \supseteq \{0, 1, \rhd, \emptyset\}$$

$\boldsymbol{\delta}$: transition function

$\boldsymbol{\delta}: [k] \times \Sigma \rightarrow Pow([k] \times \Sigma \times \{\mathbf{L}, \mathbf{R}, \mathbf{S}, \mathbf{H}\})$

many stats / symbol / dir

Deterministic (Standard) TM Execution

Definition. The **output** of the execution of a TM, $M = (\Sigma, k, \delta)$ is the result of this process:

1. Initialize $T[i] = \emptyset$ for all $i \in \mathbb{Z}$.
2. Initialize two natural number variables, $i = 0, s = 0$.
3. **repeat**
    1. $(s', \sigma', D) = \boldsymbol{\delta}(s, T[i])$
    2. $s := s', T[i] := \sigma'$
    3. if D = $\mathbf{R}$: $i := i + 1$
       if D = $\mathbf{L}$: $i := i - 1$
       if D = $\mathbf{H}$: $\mathbf{break}$
4. If the process finishes (the repeat breaks), the result of this process is $M(\cdot) = T[1], \ldots, T[m_r]$ where m_r is the smallest integer such that $\forall z > m_r . T[z] = \emptyset$. Otherwise, $M(\cdot) = \bot$.

*How should we define a **Nondeterministic Turing Machine** execution?*

Nondeterministic TM Execution

Definition. The **output** of the execution of a **TM**, $M = (\Sigma, k, \delta)$ is the result of this process:

1. Initialize $T[i] = \emptyset$ for all $i \in \mathbb{Z}$.
2. Initialize configurations, $Z = \{(T, i = 0, s = 0)\}$
3. **repeat**

    $Z' = \{\}$

    foreach $(T, i, s) \in Z$:

    foreach $(s', \sigma', D) \in \boldsymbol{\delta}(s, T[i])$

    2. $T' = T, T'[i] := \sigma'$
    3. if D = **R**: $i' := i + 1$

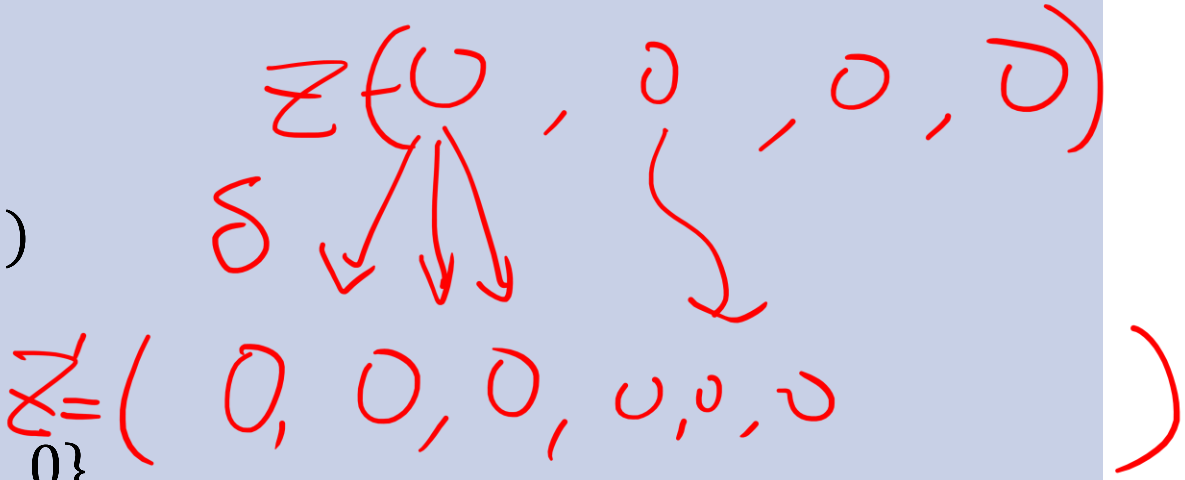
        if D = **L**: $i' := \max\{i - 1, 0\}$

        if D = **H**: **break** (outer loop)

    $Z' = Z' \cup \{(T', i', s')\}$

    $Z = Z'$

4. If the process finishes (the repeat breaks), the result of this process is $M(\cdot) = T'[1], \ldots, T'[m_r]$ where m_r is the smallest integer such that $\forall z > m_r . T'[z] = \emptyset$. Otherwise, $M(\cdot) = \bot$.



Nondeterministic TM Execution

Definition. The **output** of the execution of a **TM**, $M = (\Sigma, k, \delta)$ is the result of this process:

1. Initialize $T[i] = \emptyset$ for all $i \in \mathbb{Z}$.
2. Initialize configurations, $Z = \{(T, i = 0, s = 0)\}$
3. **repeat**

$Z' = \{\}$

foreach $(T, i, s) \in Z$:

foreach $(s', \sigma', D) \in \boldsymbol{\delta}(s, T[i])$

2. $T' = T, T'[i] := \sigma'$
3. if D = $\mathbf{R}$: $i' := i + 1$

if D = $\mathbf{L}$: $i' := \max\{i - 1, 0\}$

if D = $\mathbf{H}$: **break** (outer loop)

There are lots of other (better) ways to define the output of a nondeterministic TM, such as if any execution halts and outputs “1” the output is “1”.

4. If the process finishes (the repeat breaks), the result of this process is $M(\cdot) = T'[1], \ldots, T'[m_r]$ where m_r is the smallest integer such that $\forall z > m_r . T'[z] = \emptyset$. Otherwise, $M(\cdot) = \bot$.

Alternative definition:

A function $F: \{0,1\}^* \to \{0,1\}$ is in **NP** if there exists a non-deterministic Turing machine M and some constant $a \in \mathbb{N}^+$ such that $M(x)$ computes $F(x)$ in NTIME(n^a) for all n-bit input x.

NTIME: number of transitions from the initial to the halting state.

Equivalent to:

A function $F: \{0,1\}^* \to \{0,1\}$ is in **NP** if there exists some $a \in \mathbb{N}^+$ and $V: \{0,1\}^* \to \{0,1\}$ such that $V \in \mathbf{P}$ and $\forall x \in \{0,1\}^n, F(x) = 1 \leftrightarrow \exists w \in \{0,1\}^{n^a}$ such that $V(x, w) = 1$.

Charge

Time complexity

Class **NP**

NP-Complete

Next time:

Proof of Cook-Levin Theorem

PS10 due this Friday, Apr 25