PS10 due this Friday, Apr 25.

Class 26:
Cook-Levin Theorem

University of Virginia
cs3120: DMT2
Wei-Kai Lin

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 $x \in \{0,1\}^n$,

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

The Class P

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

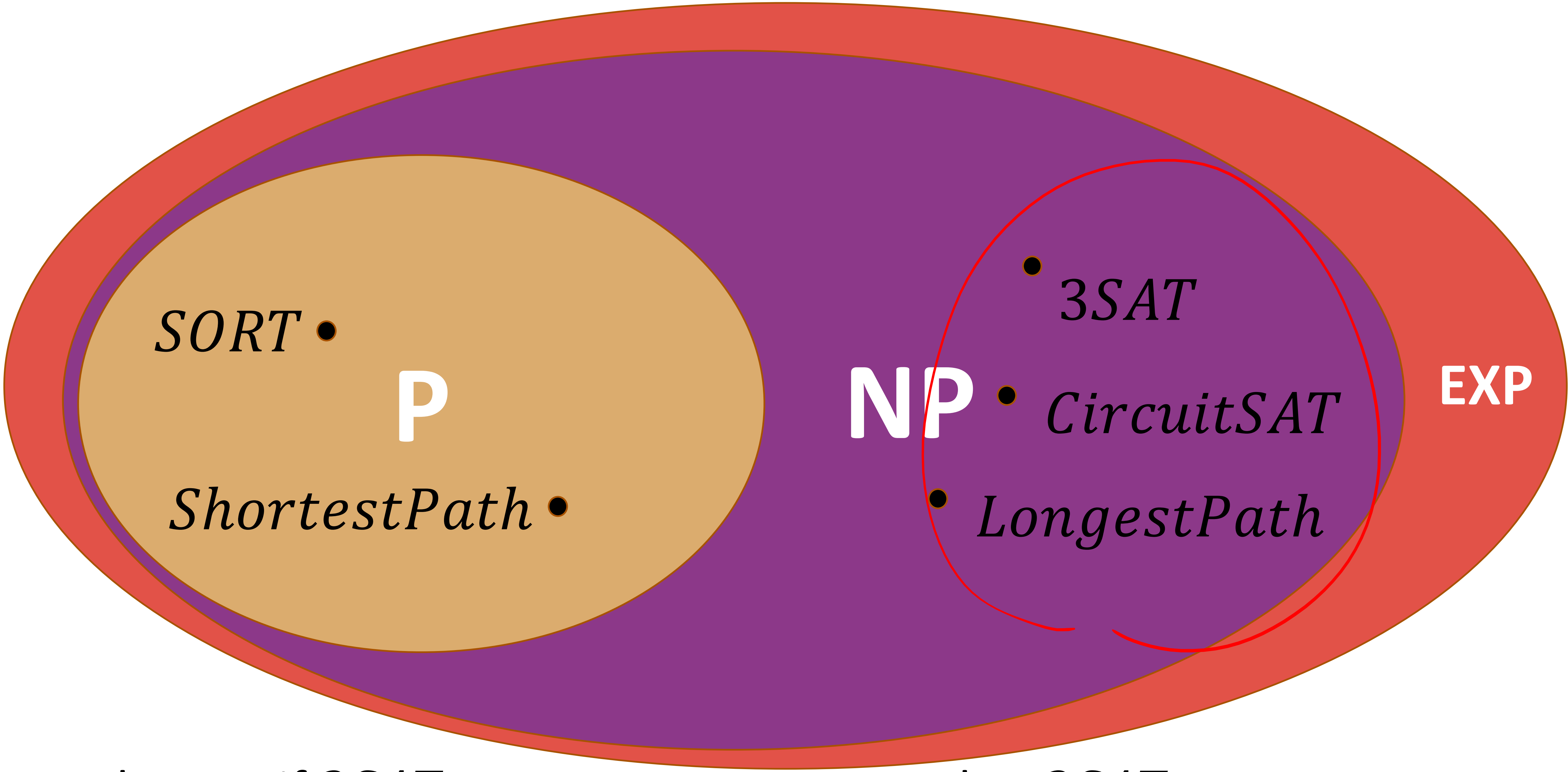
$$\bigcup_{c \in \mathbb{N}} TIME_{TM}(n^c)$$

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 $V(x,w) = 1$.



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

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

Cook-Levin Theorem

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

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

Stephen A. Cook, 1971

The Complexity of Theorem-Proving Procedures

Stephen A. Cook

University of Toronto

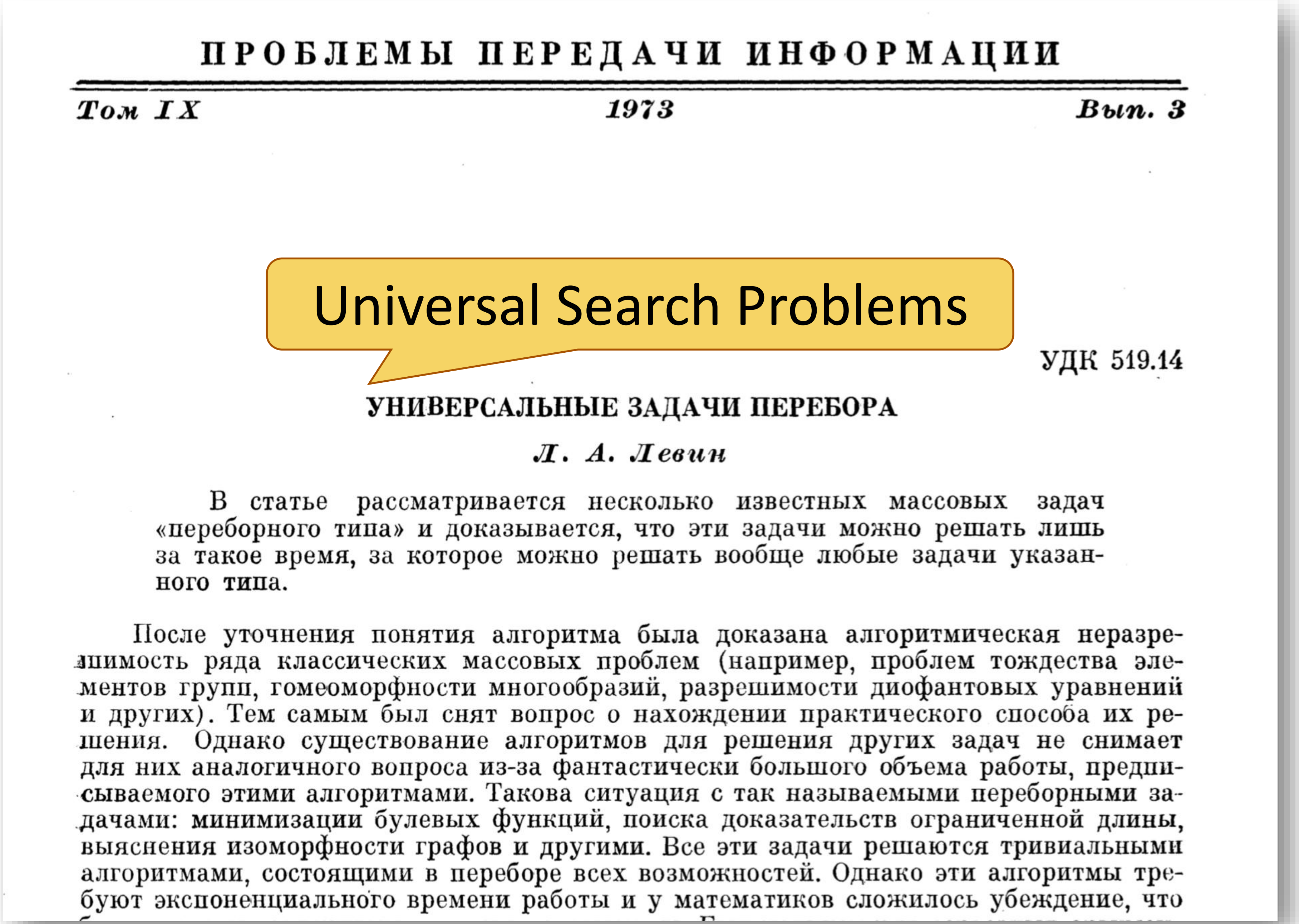
<u>Summary</u>

It is shown that any recognition problem solved by a polynomial time-bounded nondeterministic Turing machine can be "reduced" to the problem of determining whether a given propositional formula is a tautology. Here "reduced" means, roughly speaking, that the first problem can be solved deterministically in polynomial time provided an oracle is available for solving the second. From this notion of reducible, polynomial degrees of difficulty are defined, and it is shown that the problem of determining tautologyhood has the same polynomial degree as the problem of determining whether the first of two given graphs is isomorphic to a subgraph of the second. Other examples are discussed. A method of measuring the complexity of proof procedures for the predicate calculus is introduced and discussed.

certain recursive set of strings on this alphabet, and we are interested in the problem of finding a good lower bound on its possible recognition times. We provide no such lower bound here, but theorem 1 will give evidence that {tautologies} is a difficult set to recognize, since many apparently difficult problems can be reduced to determining tautologyhood. By <u>reduced</u> we mean, roughly speaking, that if tautologyhood could be decided instantly (by an "oracle") then these problems could be decided in polynomial time. In order to make this notion precise, we introduce query machines, which are like Turing machines with oracles in [1].

A <u>query machine</u> is a multitape Turing machine with a distinguished tape called the <u>query tape</u>, and three distinguished states called the <u>query state</u>, <u>yes state</u>, and <u>no</u>

Stephen Cook (2015 picture), University of Toronto



ПРОБЛЕМЫ ПЕРЕДАЧИ ИНФОРМАЦИИ

Том IX *1973* *Вып. 3*

УДК 519.14

Universal Search Problems

УНИВЕРСАЛЬНЫЕ ЗАДАЧИ ПЕРЕБОРА

Л. А. Левин

В статье рассматривается несколько известных массовых задач «переборного типа» и доказывается, что эти задачи можно решать лишь за такое время, за которое можно решать вообще любые задачи указанного типа.

После уточнения понятия алгоритма была доказана алгоритмическая неразрешимость ряда классических массовых проблем (например, проблем тождества элементов групп, гомеоморфности многообразий, разрешимости диофантовых уравнений и других). Тем самым был снят вопрос о нахождении практического способа их решения. Однако существование алгоритмов для решения других задач не снимает для них аналогичного вопроса из-за фантастически большого объема работы, предписываемого этими алгоритмами. Такова ситуация с так называемыми переборными задачами: минимизации булевых функций, поиска доказательств ограниченной длины, выяснения изоморфности графов и другими. Все эти задачи решаются тривиальными алгоритмами, состоящими в переборе всех возможностей. Однако эти алгоритмы требуют экспоненциального времени работы и у математиков сложилось убеждение, что

Leonid Levin
(born 1948)
currently at
Boston University

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**

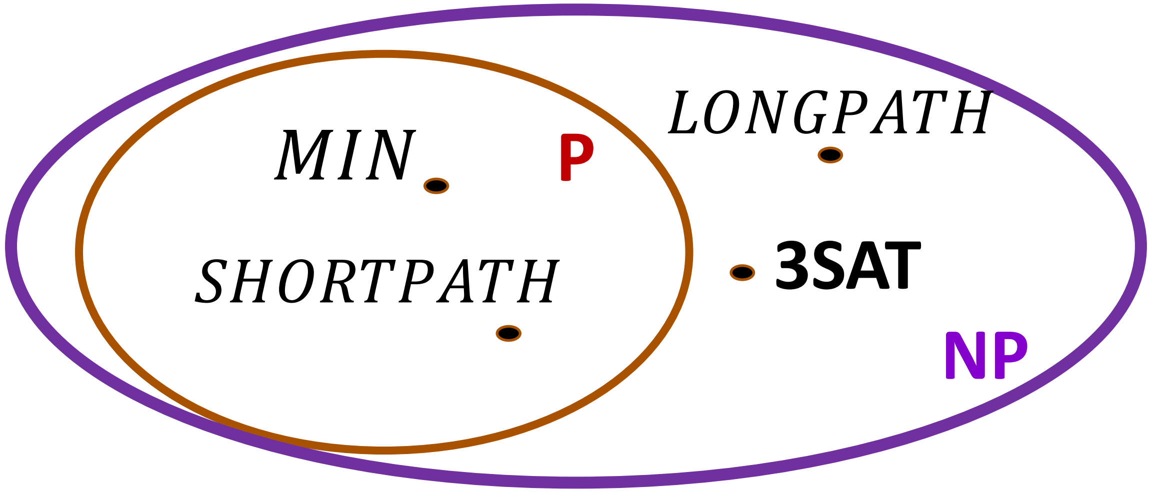
Making Progress on $\mathbf{P} \subsetneq \mathbf{NP}$

Cook-Levin Theorem: For every $F \in \mathbf{NP}$, $F \leq_p$ **3SAT.**

Equivalently: 3SAT $\in$ **$\mathbf{NP}$-Hard**

After showing 3SAT $\in \mathbf{NP}$
$\Rightarrow$ 3SAT $\in \mathbf{NP}$–Complete

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



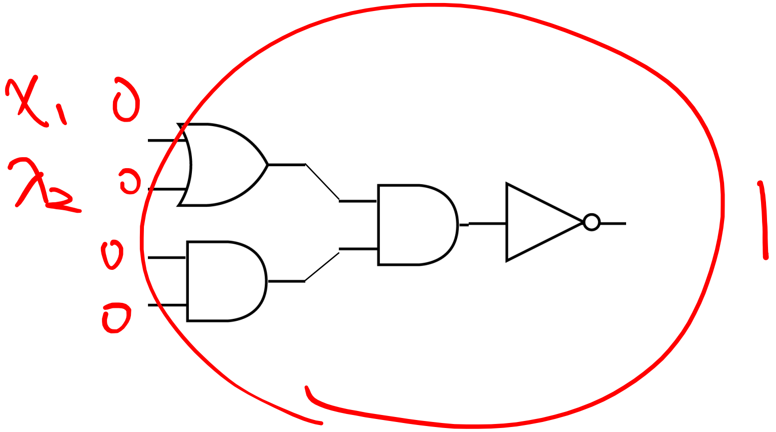
If **3SAT** $\notin$ **P**

$$\mathbf{CircuitSAT} \le_p \mathbf{NANDSAT} \le_p \mathbf{3SAT}$$

Easy: 3SAT $\le_p$ $\mathbf{CircuitSAT}$

CircuitSAT $\leq_p$ 3SAT

Suppose we are given Boolean circuit C. How do we transform it into a 3-CNF formula F such that C is satisfiable if and only if F is satisfiable?



$$F = (x_1 \vee x_2 \vee \overline{x_3}) \wedge (\ldots) \wedge \ldots$$

$x_1 = ?$

$x_2 = ?$

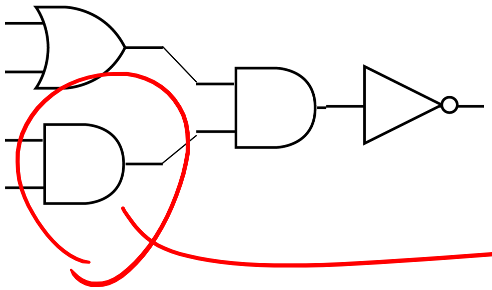
$\vdots$

CircuitSAT $\leq_p$ NANDSAT

Suppose we are given Boolean circuit C. How do we transform it into a NAND-Circuit N such that C is satisfiable if and only if N is satisfiable?

Yes: Substitute AND, OR, NOT with NAND

At most 3x more gates than C



y1=NAND(x1, x2)
y2=NAND(y1,y1)
....

NANDSAT $\le_p$ 3SAT

Suppose we are given NAND circuit C. How do we transform it into a 3-CNF formula F such that C is satisfiable if and only if F is satisfiable?

$C(\overset{=1}{x_1}, \overset{=1}{x_2})$

y1=NAND(x1, x2)

y2=NAND(y1,y1)

….

3CNF

$\underline{F(\mathrm{x})} = (x_1 \lor x_2 \lor x_3) \land (\ldots) \land \ldots.$

NANDSAT $\leq_p$ 3SAT

Useful lemma: for every gate $z = NAND(x, y)$, one can write a 3-CNF formula G_{xyz} over them such that $z = NAND(x, y)$ iff G_{xyz} is satisfiable.

$$G_{xyz} = (x \vee y \vee z) \wedge (x \vee \bar{y} \vee z) \wedge (\bar{x} \vee y \vee z) \wedge (\bar{x} \vee \bar{y} \vee \bar{z})$$

x	y	z=NAND(x,y)	G_{xyz}
0	0	1	1
0	1	1	1
1	0	1	1
1	1	0	1
0	others 0	0	0

$$G_{xyz} = (x \vee y \vee z) \wedge (x \vee \bar{y} \vee z) \wedge (\bar{x} \vee y \vee z) \wedge (\bar{x} \vee \bar{y} \vee \bar{z})$$

C
y1=NAND(x1, x2)
y2=NAND(y1,y1)
....

y_{100}

F(x1,x2,…,y1,y2,…)

$$= G_{x_1 x_2 y_1} \wedge G_{y_1 y_1 y_2} \wedge \ldots.$$

$\wedge (y_{100} \vee y_{100} \vee y_{100})$

Reduction from **NANDSAT** to 3SAT:
NANDSAT(C) = 3SAT(R (C)), where reduction R(C) = F for all C

Conclusion:

CircuitSAT $\leq_p$ **NANDSAT** $\leq_p$ 3SAT

Proof of Cook-Levin Theorem

Proving the Cook-Levin Theorem

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

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

Proof strategy:

Find a problem Z that is **NP-Hard** and show that $Z \leq_P 3SAT$.

$\forall F \in NP, \quad F \leq_p Z$

$\Rightarrow F \leq_p 3SAT$

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 $V(x,w) = 1$.

Is $CircuitSAT$ **NP-Hard** ?

We have $CircuitSAT \leq_P 3SAT$

We are finished if $CircuitSAT$ is also NP-Hard (why?)

CircuitSAT

Input: C, a string representing an n-input Boolean circuit.

Output: **1** iff there exists a string $x \in \{0, 1\}^n$ such that $C(x) = 1$.

How to prove $CircuitSAT$ is **NP-Hard** ?

CircuitSAT

Input: C, a string representing an n-input Boolean circuit.

Output: 1 iff there exists a string $x \in \{0,1\}^n$ such that $C(x) = 1$.

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

Proving $CircuitSAT$ is NP-*Hard*

Definition of Polynomial-Time Reduction $(\leq_p)$: For any $F, G\colon \{0,1\}^* \to \{0,1\}$. $F \leq_p G$ if there is a polynomial-time computable $R\colon \{0,1\}^* \to \{0,1\}^*$ such that for every $x \in \{0,1\}^*, F(x) = G(R(x))$.

$$\forall F \in \mathbf{NP}\colon\ F \leq_P CircuitSAT$$

$$\forall F \in \mathbf{NP}\colon\ \exists R \in \mathbf{P}\colon \forall x \in \{0,1\}^*\colon\ F(x) = CircuitSAT(R(x))$$

We need to show that R exists, for every function in **NP**!

Recall the Definition of Class 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 $V(x, w) = 1$.

Definition of $\leq_p$: For any $F, G\colon \{0,1\}^* \to \{0,1\}$. $F \leq_p G$ if there is a p-time $R\colon \{0,1\}^* \to \{0,1\}^*$ such that for every $x \in \{0,1\}^*$, $F(x) = G(R(x))$.

Definition of **NP**: $F \in \mathbf{NP}$ if $\exists\, V \in \mathbf{P}$ such that $\forall x \in \{0,1\}^n, \exists\, w \in \{0,1\}^{n^a}$ such that $F(x) = 1$ iff $V(x,w) = 1$.

Goal: $\forall F \in \mathbf{NP}\colon\ F \leq_P CircuitSAT$

$$\forall F \in \mathbf{NP}\colon\ \exists R \in \mathbf{P}\colon \forall x \in \{0,1\}^*\colon\ F(x) = CircuitSAT(R(x))$$

$$\exists V \in \mathbf{P}, a \in \mathbb{N}^+\colon$$
$$\forall x \in \{0,1\}^n, \exists w \in \{0,1\}^{n^a} \text{such that } F(x) = 1 \text{ iff } V(x,w) = 1.$$

Definition of $\leq_p$: For any F, G: $\{0,1\}^* \to \{0,1\}$. $F \leq_p G$ if there is a p-time R: $\{0,1\}^* \to \{0,1\}^*$ such that for every $x \in \{0,1\}^*$, $F(x) = G(R(x))$.

Definition of **NP**: $F \in \mathbf{NP}$ if $\exists\, V \in \mathbf{P}$ such that $\forall x \in \{0,1\}^n, \exists\, w \in \{0,1\}^{n^a}$ such that $F(x) = 1$ iff $V(x,w) = 1$.

Goal: $\forall F \in \mathbf{NP}$: $F \leq_P CircuitSAT$

$$\forall F \in \mathbf{NP}:\ \exists R \in \mathbf{P}: \forall x \in \{0,1\}^*:\ F(x) = CircuitSAT(R(x))$$

$$\exists V \in \mathbf{P}, a \in \mathbb{N}^+:$$

$$\forall x \in \{0,1\}^n, \exists w \in \{0,1\}^{n^a} \text{such that } F(x) = 1 \text{ iff } V(x,w) = 1.$$

There exists a Boolean circuit $C_x(\star)$ that computes $V(x, \star)$:

$C_x(*)$ { Input: $\star \in \{0,1\}^{n^a}$ length = n^a
Output: $V(x, \star)$

C_x is SAT iff $F(x) = 1$

$= CircSAT(C_x) = 1$

How do we know C_x exists?

$$\exists V \in \mathbf{P}, a \in \mathbb{N}^+:$$
$$\forall x \in \{0,1\}^n, \exists w \in \{0,1\}^{n^a} \text{such that } F(x) = 1 \text{ iff } V(x,w) = 1.$$

There exists a Boolean circuit $C_x(\star)$ that computes $V(x, \star)$:

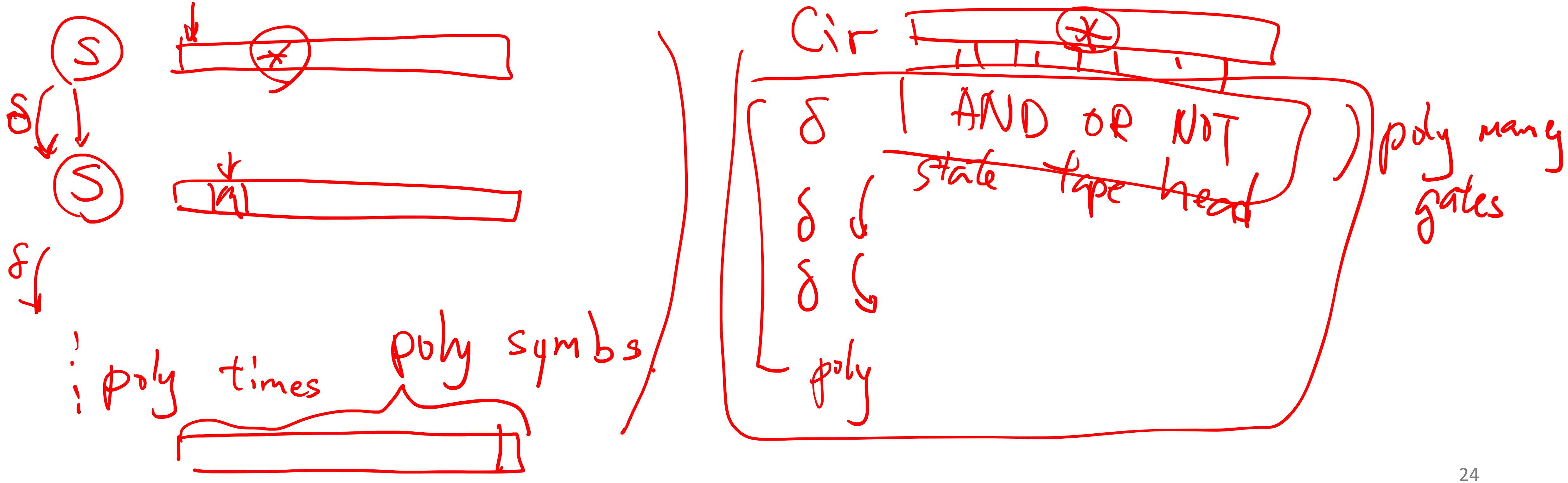
Input: $\star \in \{0,1\}^{n^a}$ length = n^a

Output: $V(x, \star)$

$C_x(w) = 1$

Boolean circuit $C_x(\star)$ that computes $V(x,\star)$

$V \in \mathbf{P}$ so we have TM M computes V in poly-time.
Fixing x, we also have M_x computes $V(x, \ \star)$ in poly-time.
We want C_x that simulates M_x and $|C_x| = poly(|x|)$.



Boolean circuit $C_x(\star)$ that computes $V(x,\star)$

$V \in \mathbf{P}$ so we have TM M computes V in poly-time.
Fixing x, we also have M_x computes $V(x,\ \star)$ in poly-time.
We want C_x that <u>simulates</u> M_x and $|\ C_x\ | = poly(|x|)$.

Theorem:
For any $f \in \mathbf{P}$, (f is computable by <u>TM</u> in poly time)
$f \in \mathbf{P}_{/\mathbf{poly}} = \bigcup_{c\in\mathbb{N}} SIZE(n^c)$ (f is computable by <u>circuit</u> in poly size)

Theorem 13.12 (Non-uniform computation contains uniform computation)
There is some $a \in \mathbb{N}$ s.t. for every nice $T : \mathbb{N} \to \mathbb{N}$ and $F : \{0,1\}^* \to \{0,1\}$,

$$TIME(T(n)) \subseteq SIZE(T(n)^a)\ .$$

Completing the Proof

Cook-Levin Theorem: For every $F \in \mathbf{NP}$, $F \leq_p$ **3SAT.**

We have proven *CircuitSAT* is **NP-Hard**:

$$\forall F \in \mathbf{NP}: F \leq_p CircuitSAT$$

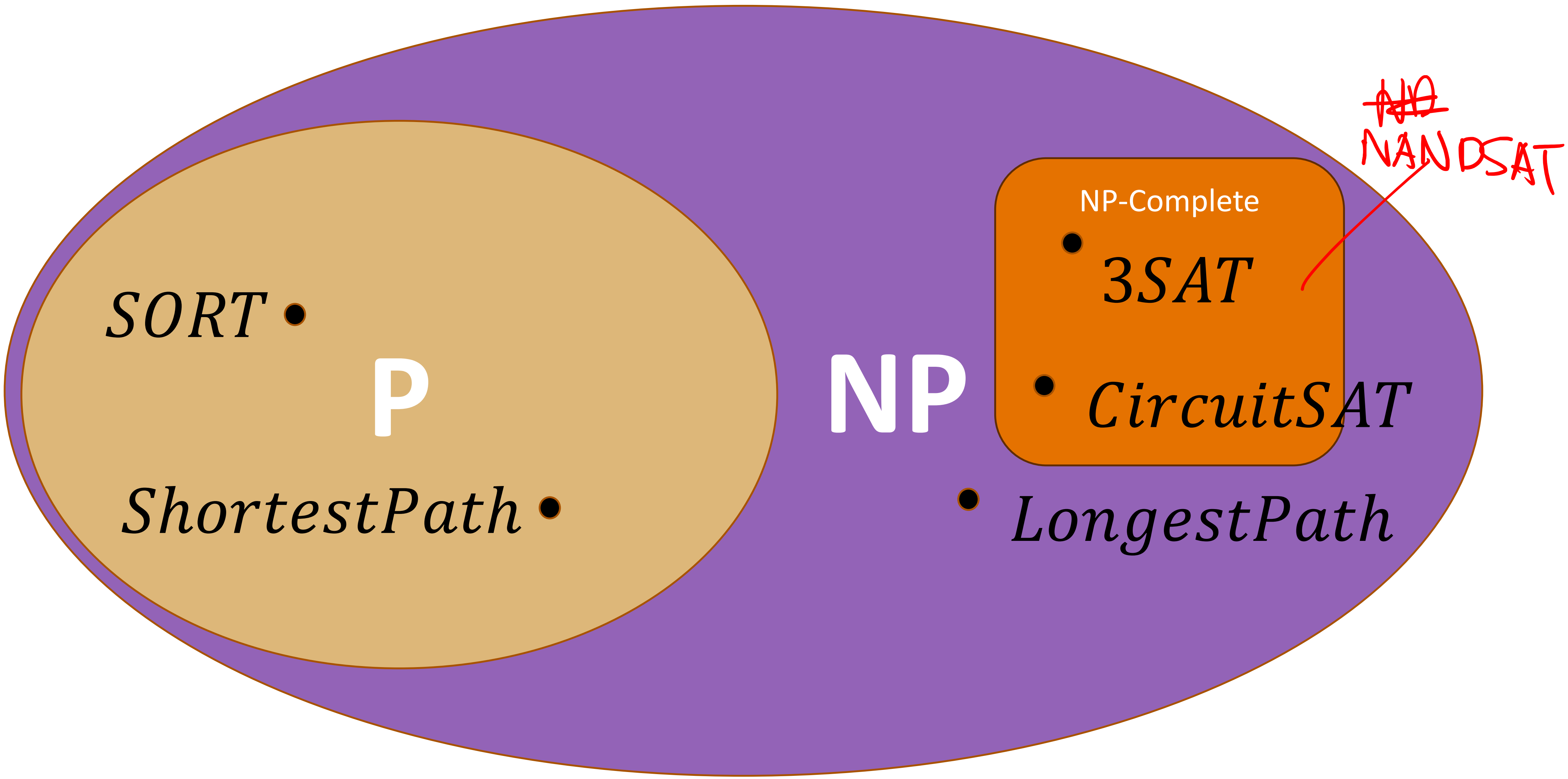
Cook-Levin Theorem: For every $F \in \mathbf{NP}$, $F \leq_p$ **3SAT.**

We have proven *CircuitSAT* is **NP-Hard**:

$$\forall F \in \mathbf{NP}: F \leq_p CircuitSAT$$

$$F \leq_p CircuitSAT \leq_p 3SAT$$

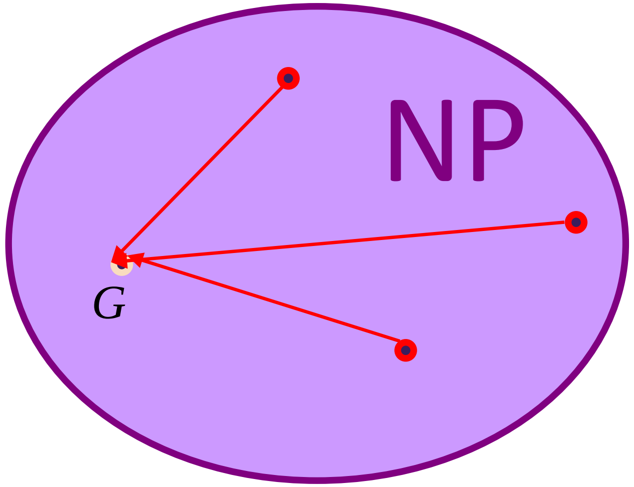
Thus, if there is a polynomial time algorithm for 3SAT, there is a polynomial time algorithm for *every* problem in **NP**!

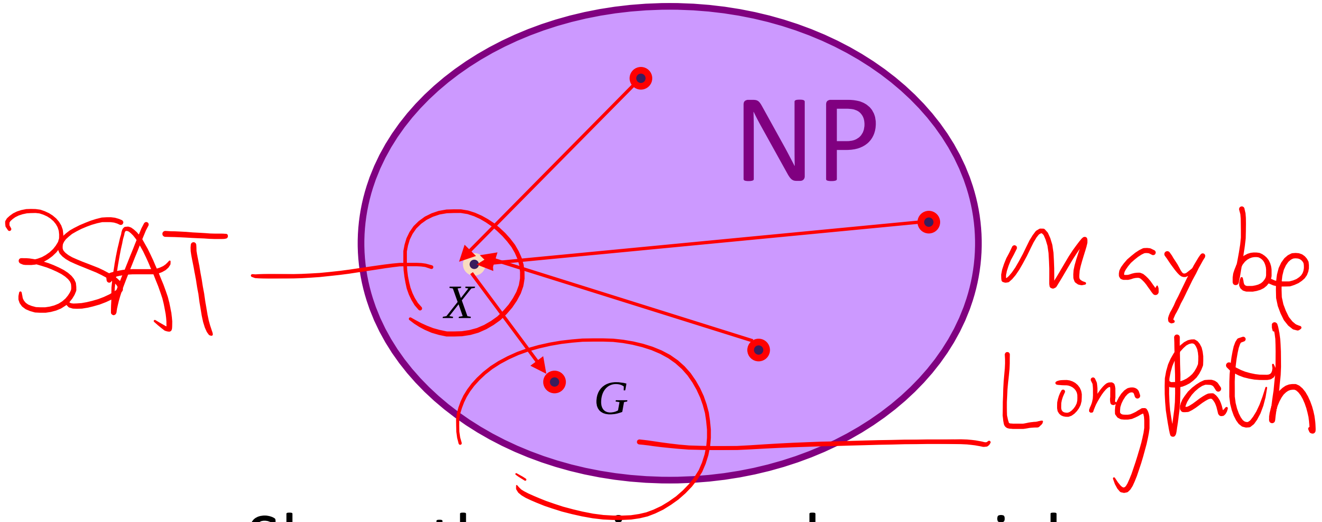


Unknown if $3SAT \in$ **P** Known that $3SAT \in$ **NP**

More NP-Complete problems

Proving G is NP-Hard





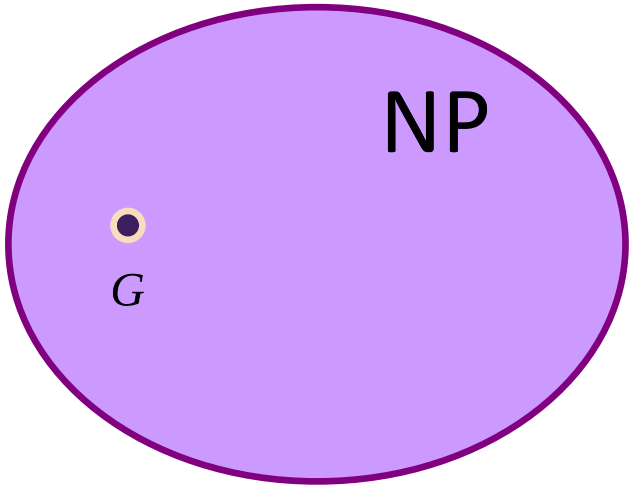
Show there is a polynomial-time reduction from every problem $F \in \mathbf{NP}$ to G.

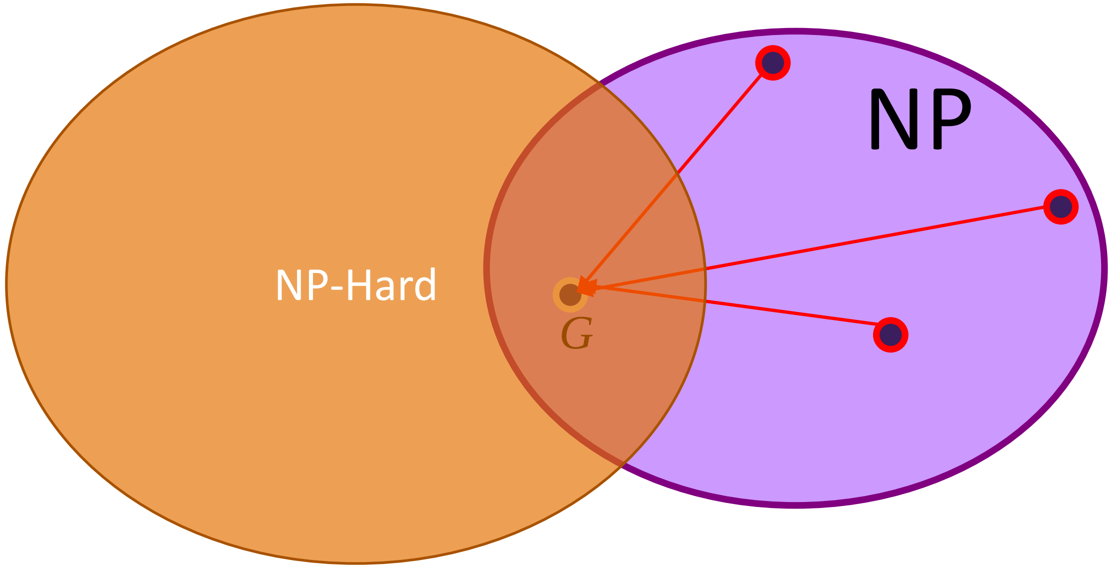
$$\forall F \in \mathbf{NP}: F \leq_p G$$

Show there is a polynomial-time reduction from **one problem** $X \in$ **NP-Hard** to G.

$$\exists X \in \textbf{NP–Hard}: X \leq_p G$$

Proving G is NP-Complete



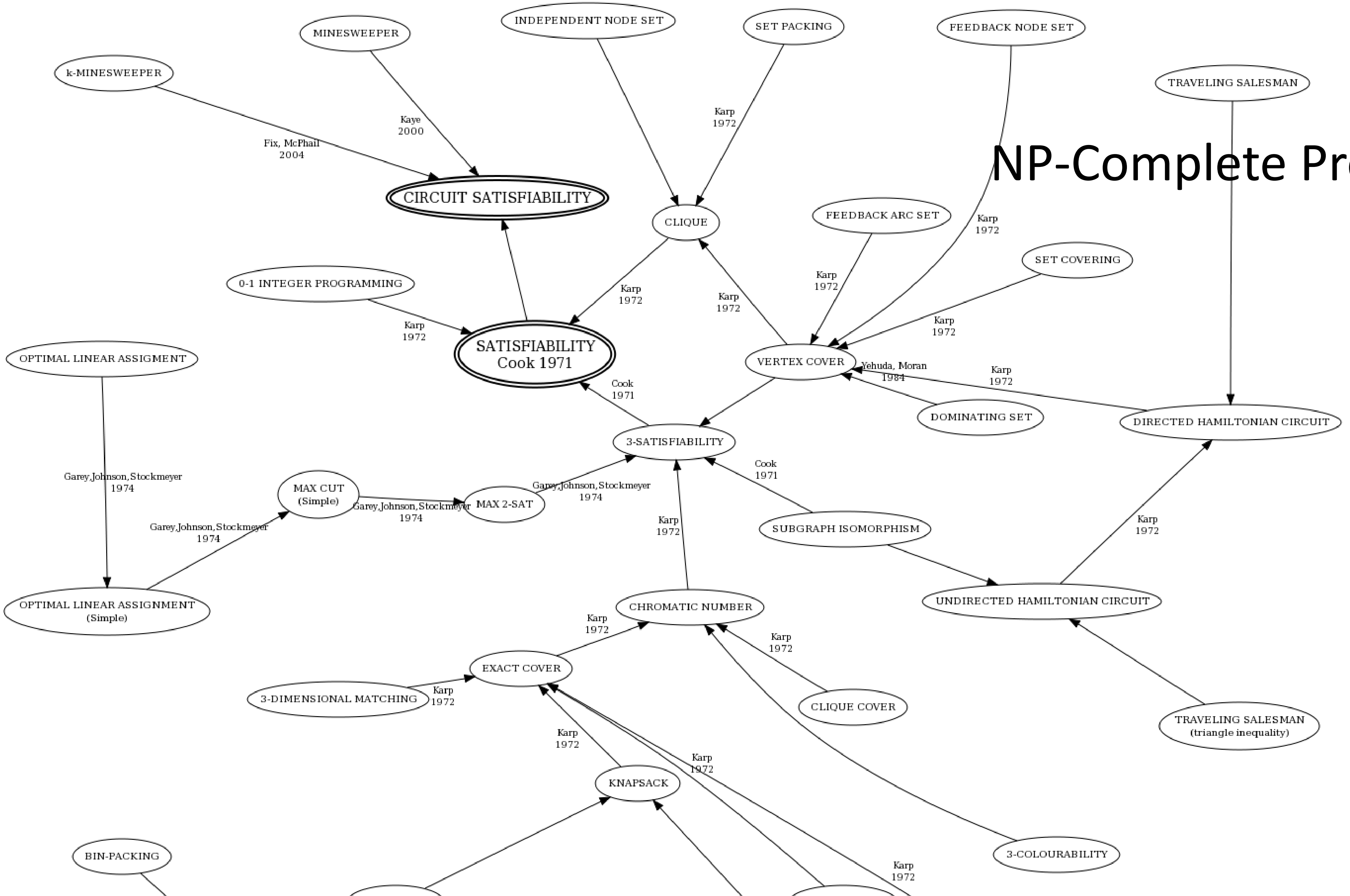


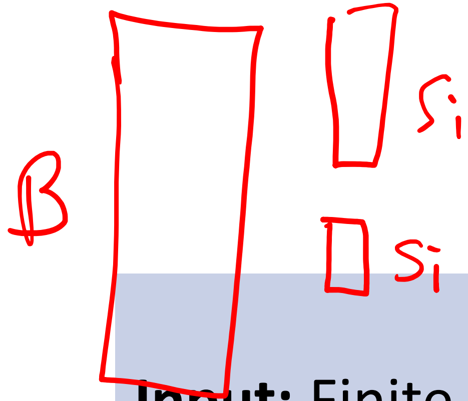
1. $G \in \mathbf{NP}$

2. $G \in$ **NP–Hard**

$\exists X \in$ **NP–Hard**: $X \leq_p G$

NP-Complete Problems





Some NP-Complete Problems

k

BinPacking

Input: Finite set of items, I, size $s(i)$ for each item, bin capacity B, number of items k

Output: 1 iff there is a way to pack $\geq k$ items in B

Visitor (a.k.a., Traveling Seller Problem)

Input: A weighted graph $G = (V, E), w(e \in E) \rightarrow \mathbb{N}$, a start node s $\in V$, a cost z

Output: 1 iff there path in G that visits every node with code $\leq z$

SetCover

Input: A set $U, S \subseteq Pow(U)$ where $\bigcup_{s \in S} s = U$, $m \in \mathbb{N}$

Output: 1 iff there is a subset S' of S where $\bigcup_{s \in S'} s = U$ and $|S'| \leq m$.

Which outcome do we prefer?

factoring ~~∉~~ ∈? P

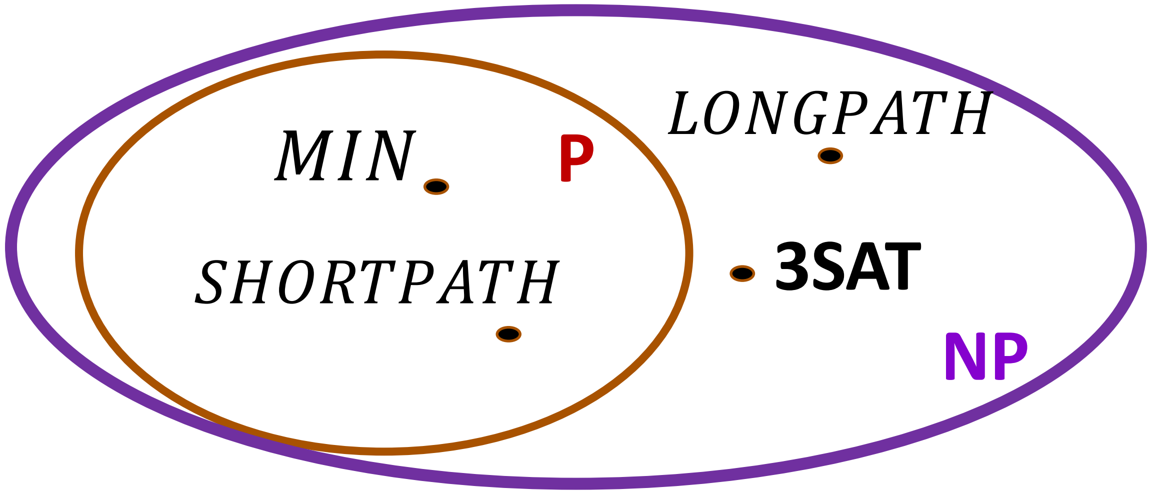
P = NP

∈ NP

3SAT ∈ P

✓

P ⊊ NP



3SAT ∉ P

Do we “solve” these okay today?

BinPacking

Input: Finite set of items, I, size $s(i)$ for each item, bin capacity B, number of items k

Output: 1 iff there is a way to pack $\geq k$ items in B

Visitor (a.k.a., Traveling Seller Problem)

Input: A weighted graph $G = (V, E), w(e \in E) \rightarrow \mathbb{N}$, a start node s $\in V$, a cost z

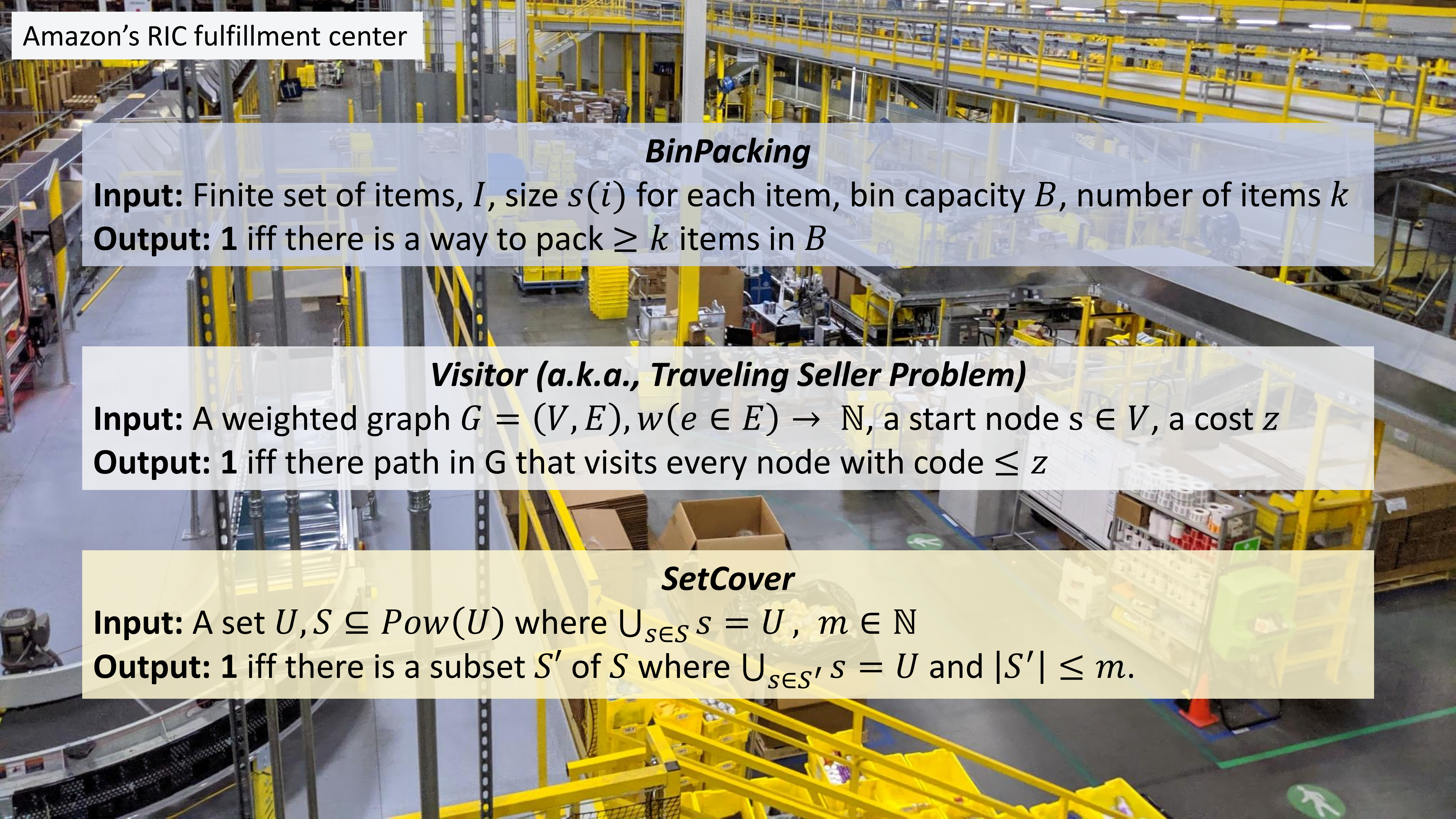
Output: 1 iff there path in G that visits every node with code $\leq z$

SetCover

Input: A set $U, S \subseteq Pow(U)$ where $\bigcup_{s \in S} s = U$, $m \in \mathbb{N}$

Output: 1 iff there is a subset S' of S where $\bigcup_{s \in S'} s = U$ and $|S'| \leq m$.

Amazon’s RIC fulfillment center



Amazon’s RIC fulfillment center

BinPacking

Input: Finite set of items, I, size $s(i)$ for each item, bin capacity B, number of items k

Output: 1 iff there is a way to pack $\geq k$ items in B

Visitor (a.k.a., Traveling Seller Problem)

Input: A weighted graph $G = (V, E), w(e \in E) \rightarrow \mathbb{N}$, a start node s $\in V$, a cost z

Output: 1 iff there path in G that visits every node with code $\leq z$

SetCover

Input: A set $U, S \subseteq Pow(U)$ where $\bigcup_{s \in S} s = U,\ m \in \mathbb{N}$

Output: 1 iff there is a subset S' of S where $\bigcup_{s \in S'} s = U$ and $|S'| \leq m$.

What about 3SAT?

50 variables, 100 random clauses

sahands / simple-sat

Watch 5 | Unstar 29 | Fork 5

Code | Issues 0 | Pull requests 0 | Wiki | Pulse | Graphs

Simple recursive and iterative SAT solver written in Python.

38 commits | 1 branch | 0 releases | 1 contributor

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

sahands Moving article content. Latest commit b987fb0 on Sep 30, 2014

src	Added the post content to the repository.	2 years ago
.gitignore	Initial commit	2 years ago
LICENSE	Initial commit	2 years ago
README.rst	Renames	2 years ago

README.rst

sim

This p
based

```
> time python3 sat.py -i random_50_100.in
~x48 ~x4 ~x9 ~x37 ~x44 ~x50 ~x1 ~x28 ~x8 ~x21 ~x27 ~x32 ~x17 ~x29 ~x30 x24 ~x22
~x25 ~x14 ~x15 x45 ~x16 x36 ~x26 ~x33 ~x5 x18 x7 ~x31 x38 ~x49 ~x6 x34 x46 ~x35
x39 ~x43 x40 ~x19 ~x42 ~x20 ~x10 x12 x23 ~x3 x13 x47 ~x11 ~x41 ~x2

real    0m0.053s
user    0m0.042s
sys     0m0.009s
>
```

If knowing **P=NP** *wouldn't help us much in solving practical problems we want to solve, would there be a benefit to knowing* **P $\subsetneq$ NP?**

Example: Encryption

Some details needed:
Key-length < message
Correct decryption

AES (Advanced Encryption Standard) is a poly-time function:
AES(key, message) $\mapsto$ ciphertext

prob

Fix message = "secret Amazon order ….", define function:

$C(x) =$ 1 if exists k s.t. AES(k, message) = x
0 otherwise

We have $C \in NP$ (why?)

W = key

If P= NP, what happened to C?

Hard Problems are Useful

P = NP would imply most cryptographic goals are impossible: no public-key cryptosystem could exist!

Assumptions in Cryptography:

Factoring is **hard** $\Longrightarrow$ **RSA** is secure

NP-Hard problems are **hard**
$\Longrightarrow$ some "post quantum" cryptosystems are secure

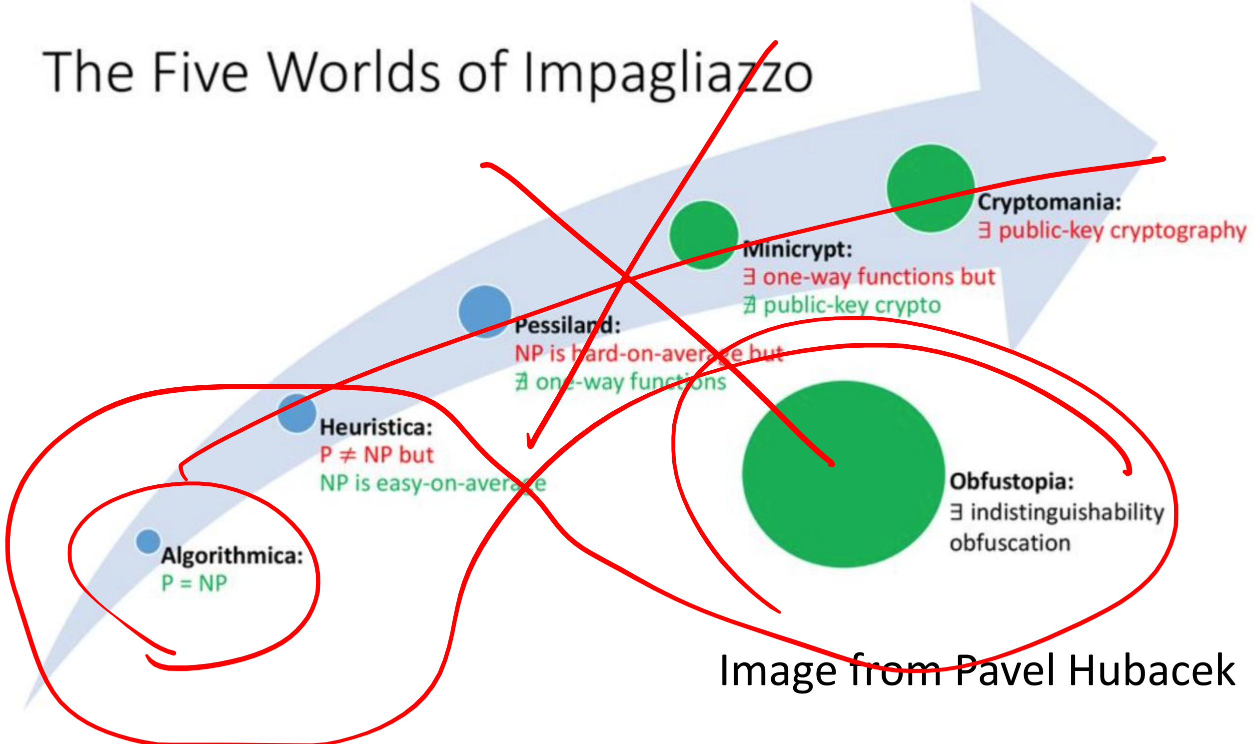


Image from Pavel Hubacek

Charge

Time complexity

Cook-Levin Theorem

PS10 due this Friday, Apr 25