Hilbert's Problems and Complexity

- In 1900, mathematician David Hilbert delivered a now-famous address at the International Congress of Mathematicians in Paris.
- Identified twenty-three mathematical problems and posed them as a challenge for the coming century.
- Hilbert's tenth problem was to devise an algorithm that tests whether a polynomial has an integral root. [Some polynomials have an integral root and some do not.]
- Hilbert's tenth problem asks in essence whether the set D is decidable.

$$D = \{p \mid p \text{ is a polynomial with an integral root}\}.$$

Hilbert's Problems and Complexity

- For single variable:

$$D_1 = \{p \mid p \text{ is a polynomial over } x \text{ with an integral root}\}.$$

Here is a TM M_1 that recognizes D_1:

M_1 = "The input is a polynomial p over the variable x.

1. Evaluate p with x set successively to the values 0, 1, −1, 2, −2, 3, −3, ... If at any point the polynomial evaluates to 0, *accept*."

- If p has an integral root, *M1* eventually will find it and accept. If p does not have an integral root, *M1* will run forever.

- For single variable, bound exist. $\pm k \frac{c_{\max}}{c_1}$,

- For multivariable, no such bound exist.

Complexity Classes

$\mathsf{P} \equiv \mathsf{DTIME}(\mathsf{poly}(n)) \equiv \bigcup_{k>0} \mathsf{DTIME}(n^k)$

$\mathsf{NP} \equiv \mathsf{NTIME}(\mathsf{poly}(n))$

$\mathsf{EXP} \equiv \bigcup_{k>0} \mathsf{DTIME}(2^{n^k})$

$\mathsf{NEXP} \equiv \bigcup_{k>0} \mathsf{NTIME}(2^{n^k})$

$\mathsf{LOG} \equiv \mathsf{DSPACE}(\log(n))$

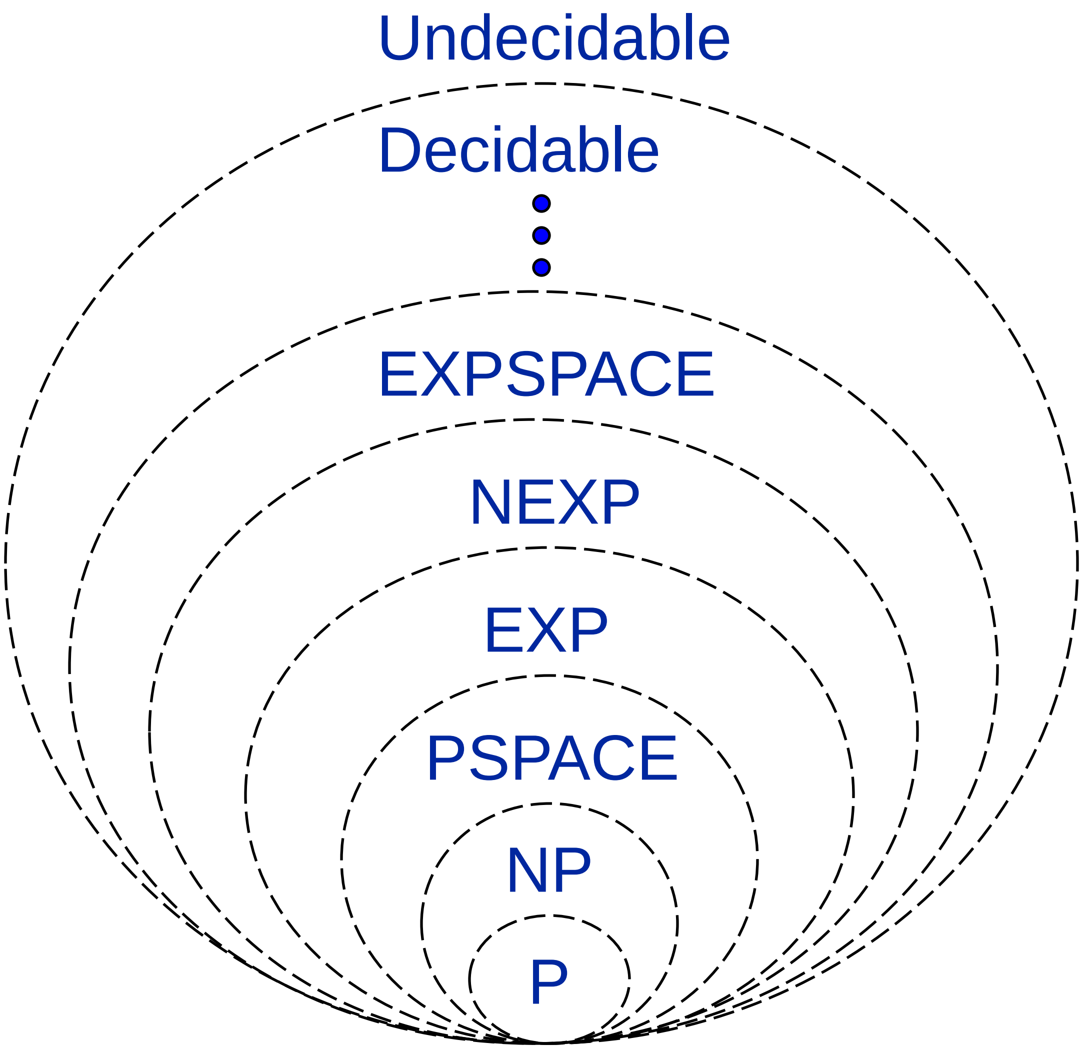
$\mathsf{NLOG} \equiv \mathsf{NSPACE}(\log(n))$

$\mathsf{PSPACE} \equiv \mathsf{DSPACE}(\mathsf{poly}(n))$

$$\mathsf{LOG} \subseteq \mathsf{NLOG} \subseteq \mathsf{P} \subseteq \mathsf{NP} \subseteq \mathsf{PSPACE} \subseteq \mathsf{EXP} \subseteq \mathsf{NEXP}.$$

- NL $\subseteq$ P $\subseteq$ NP $\subseteq$ PSPACE = NPSPACE $\subseteq$ EXPTIME
- P $\subset$ EXPTIME
- NL $\subset$ PSPACE

Hierarchy of Complexity Classes



PSPACE $\subset$ EXPSPACE

P $\subset$ EXP

PSPACE = NPSPACE

P $\subseteq$ NP $\subseteq$ PSPACE

P =? NP

Complexity Class P

$$\begin{aligned} \text{CONNECTED} &:= \{\langle G\rangle \mid G \text{ is a connected undirected graph}\} \\ \text{BIPARTITE} &:= \{\langle G\rangle \mid G \text{ is an undirected bipartite graph}\} \\ \text{TRIANGLE-FREE} &:= \{\langle G\rangle \mid G \text{ is a triangle-free undirected graph}\} \end{aligned}$$

$$\text{PATH} := \{\langle G, s, t\rangle \mid \text{ There is a path from vertex } s \text{ to vertex } t \text{ in a directed graph } G\}$$

$$\text{RELPRIME} := \{\langle x, y\rangle \mid \text{ The positive integers } x \text{ and } y \text{ are relatively prime}\}$$

Complexity Class NP

The class $\mathbf{NP}$

Class of problems having efficiently verifiable solutions.

A decision problem/language is in $\mathbf{NP}$ if given an input x, we can easily verify that x is a YES instance of the problem (x is in the language) if we are given the polynomial-size solution for x, that certifies this fact.

Def: A language $L \subseteq \{0,1\}^*$ is in $\mathbf{NP}$ if there exists a polynomial p and a polynomial-time Turing machine M such that for every $x \in \{0,1\}^n$:

$$x \in L \;\Leftrightarrow\; \exists u \in \{0,1\}^{p(|x|)} \colon M(x,u) = 1 \;.$$

If $x \in L$ and $u \in \{0,1\}^{p(|x|)}$ satisfy $M(x,u) = 1$ then we call u a *certificate* (or a *witness*) for x (with respect to the language L and machine M).

Complexity Class NP

Relation between NP and P

We have the following trivial relationships between NP and the classes P and $\mathrm{DTIME}(T(n))$:

Claim 2.3: $\mathrm{P} \subseteq \mathrm{NP} \subseteq \bigcup_{c>1} \mathrm{DTIME}(2^{n^c})$.

Proof: Suppose $L \in \mathrm{P}$ is decided in poly-time by M, i.e.

$$x \in L \Leftrightarrow M(x) = 1 \Leftrightarrow \exists u \in \{0,1\}^0 M(x,u) = 1 \ .$$

Hence, $L \in \mathrm{NP}$.

If $L \in \mathrm{NP}$ and M and, $p(n)$ are as in the definition of NP, then we can decide L in time $2^{O(p(n))}$ by enumerating all possible u and using M to check whether u is a valid certificate for the input x. The machine accepts iff such a u is ever found. Since $p(n) = O(n^c)$ for some $c > 1$, then this machine runs in $2^{O(n^c)}$ time.

Complexity Class NP

Non-deterministic Turing machines

The class $\mathbf{NP}$ can also be defined using non-deterministic Turing machines (NDTMs). The only differences between an NDTM and a TM are:

- NDTM has two transition functions δ_0 and δ_1.
- NDTM has a special state we denote by q_{accept}.
- NDTM makes (at each step) an arbitrary choice as to which of its two transition functions to apply.

We say that a NDTM N outputs 1 on a given input x if there is some sequence of these non-deterministic choices that would make N reach q_{accept} on input x. Otherwise, if every sequence of choices makes N halt without reaching q_{accept}, then we say that N outputs 0.

We say that N runs in $T(n)$ time if for every $x \in \{0,1\}^n$ and every sequence of choices, $M(x)$ reaches either the halting state or q_{accept} within $T(|\, x \,|)$ steps.

Complexity Class NP

Alternative definition of NP

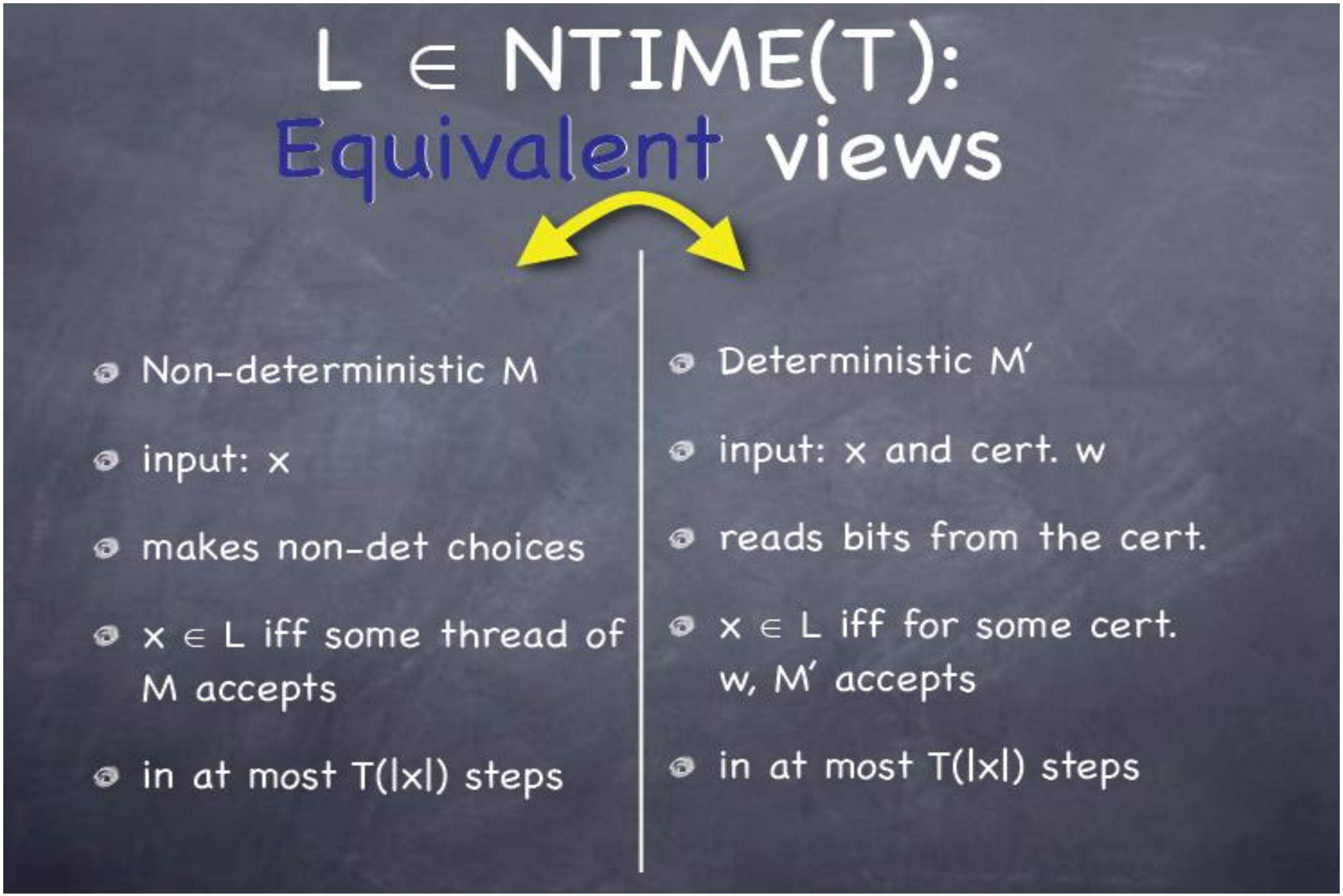
Def: For every function $T : \mathbb{N} \to \mathbb{N}$ and $L \subseteq \{0,1\}^*$, we say that $L \in \mathrm{NTIME}(T(n))$ if there is a constant $c > 0$ and a $cT(n)$-time NDTM N such that for every $x \in \{0,1\}^n$: $x \in L \Leftrightarrow N(x) = 1$.

Theorem 2.6: $\mathrm{NP} = \cup_{c \in \mathbb{N}} \mathrm{NTIME}(n^c)$.

Proof idea: If L is decided by a $p(n)$-time NDTM N, then the sequence of choices that lead to q_{accept} can be used as a certificate of size $p(n)$.

If $L \in \mathrm{NP}$ (with machine M and cert-size $p(n)$) then we can construct a NDTM N that given $x \in \{0,1\}^n$ as input first makes $p(n)$ non-deterministic choices to write down $u \in \{0,1\}^{p(n)}$; after that, N computes $M(x,u)$ and finishes in state q_{accept} if $M(x,u) = 1$, otherwise N just halts.

Complexity Class NP



NP: examples

Problems in $\mathbf{NP}$

Independent set: Given a graph G and a number k, decide if there is a k-size independent subset of vertices in G. The certificate is the list of k vertices forming an independent set.

Traveling salesman: Given a set of n nodes, $\binom{n}{2}$ numbers d_{ij} denoting the distances between all pairs of nodes, and a number k, decide if there is a closed circuit (i.e., a "salesman tour") that visits every node exactly once and has total length at most k. The certificate is the sequence of nodes in the tour.

Subset sum: Given a list of n numbers $A_1, \ldots, A_n$ and a number T, decide if there is a subset of the numbers that sums up to T. The certificate is the list of members in this subset.

NP: examples

Problems in $\mathbf{NP}$

Linear programming: Given a list of m linear inequalities with rational coefficients over n variables $u_1, \ldots, u_n$ (in the form $a_1u_1 + a_2u_2 + \ldots + a_nu_n \leq b$ for some coefficients $a_1, \ldots, a_n, b$), decide if there is an assignment of rational numbers to the variables $u_1, \ldots, u_n$ that satisfies all the inequalities. The certificate is the assignment.

Integer programming: Given a list of m linear inequalities with rational coefficients over n variables $u_1, \ldots, u_m$, find out if there is an assignment of integer numbers to $u_1, \ldots, u_n$ satisfying the inequalities. The certificate is the assignment.

Graph isomorphism: Given two $n \times n$ adjacency matrices M_1 and M_2, decide if M_1 and M_2 define the same graph, up to renaming of vertices. The certificate is the permutation $\pi : [n] \to [n]$, such that M_2 is equal to M_1 after reordering M_1s indices according to π.

NP: examples

Problems in NP

Composite numbers: Given a number N decide if N is a composite (i.e., non-prime) number. The certificate is the factorization of N.

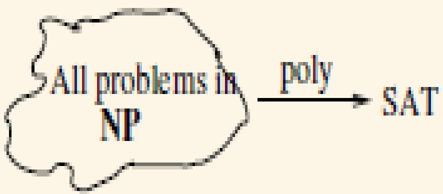
Factoring: Given three numbers N, L and U decide if N has a factor M in the interval $[L, U]$. The certificate is the factor M.

Connectivity: Given a graph G and two vertices s, t in G, decide if s is connected to t in G. The certificate is the path from s to t.

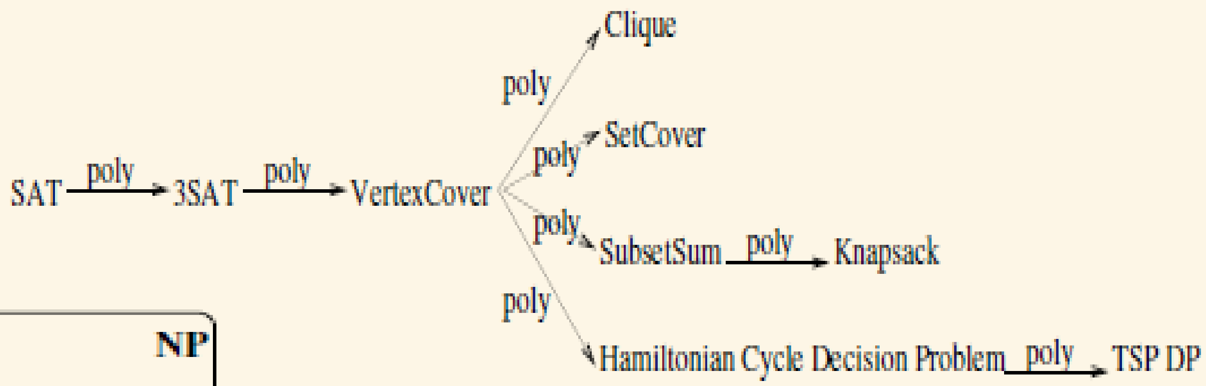
NP: examples

$$
\begin{aligned}
\text{HAMPATH} &:= \{\langle G, s, t\rangle \mid \text{There is a Hamiltonian path from vertex } s \text{ to vertex } t \text{ in the directed graph } G\} \\
\text{UHAMPATH} &:= \{\langle G, s, t\rangle \mid \text{There is a Hamiltonian path from vertex } s \text{ to vertex } t \text{ in the undirected graph } G\} \\
\text{CLIQUE} &:= \{\langle G, k\rangle \mid \text{The undirected graph } G \text{ has a } k\text{-clique}\} \\
\text{INDEP-SET} &:= \{\langle G, k\rangle \mid \text{The undirected graph } G \text{ has an independent set of size } k\} \\
\text{VERTEX-COVER} &:= \{\langle G, k\rangle \mid \text{The undirected graph } G \text{ has a vertex cover of size } k\} \\
\text{COMPOSITE} &:= \{\langle x\rangle \mid \text{The positive integer } x \text{ is composite}\} \\
\text{SUBSET-SUM} &:= \Big\{\langle S, t\rangle \mid \text{There is a subset } T \text{ of the set } S \text{ with } t = \sum_{x \in T} x\Big\} \\
\text{SAT} &:= \{\langle \phi\rangle \mid \phi \text{ is a satisfiable Boolean formula}\} \\
\text{3SAT} &:= \{\langle \phi\rangle \mid \phi \text{ is a satisfiable Boolean formula in 3-cnf}\}
\end{aligned}
$$

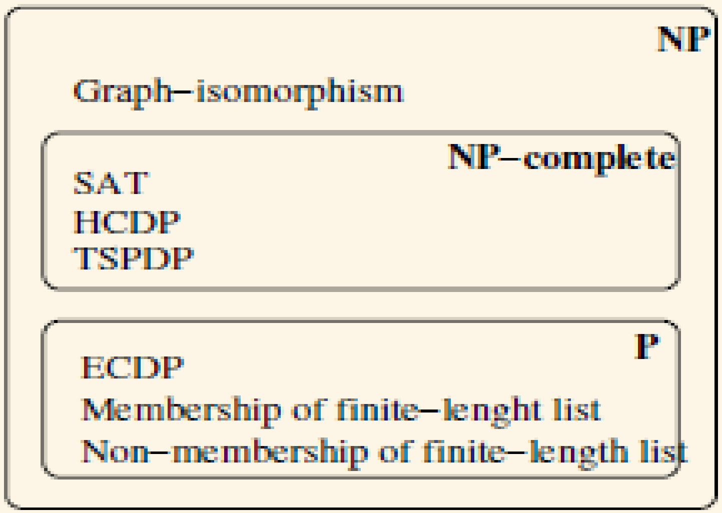
NP Complete



All problems in
NP
poly
SAT



SAT
poly
3SAT
poly
VertexCover
poly
Clique
poly
SetCover
poly
SubsetSum
poly
Knapsack
poly
Hamiltonian Cycle Decision Problem
poly
TSP DP



NP
Graph–isomorphism
NP–complete
SAT
HCDP
TSPDP
P
ECDP
Membership of finite–lenght list
Non–membership of finite–length list

P & NP-Complete Problems

- **Shortest simple path**
  - Given a graph G = (V, E) find a **shortest** path from a source to all other vertices
  - <u>Polynomial solution:</u> O(VE)
- **Longest simple path**
  - Given a graph G = (V, E) find a **longest** path from a source to all other vertices
  - <u>NP-complete</u>

P & NP-Complete Problems

- **3-CNF** is NP-Complete
- Interestingly enough, **2-CNF** is in P!

P & NP-Complete Problems

- **Euler tour**
  - G = (V, E) a connected, directed graph find a cycle that traverses each edge of G exactly once (may visit a vertex multiple times)
  - Polynomial solution O(E)
- **Hamiltonian cycle**
  - G = (V, E) a connected, directed graph find a cycle that visits each vertex of G exactly once
  - NP-complete

Complexity Class coNP

$\mathbf{coNP}$

Def: $\mathrm{coNP} = \{L \subseteq \{0,1\}^* : \overline{L} \in \mathrm{NP}\}$.

Hence, $\overline{\text{SAT}} \in \mathrm{coNP}$.

Alternative def:

For every $L \subseteq \{0,1\}^*$, we say that $L \in \mathbf{coNP}$ if there exists a polynomial $p : \mathbb{N} \to \mathbb{N}$ and a polynomial-time TM M such that for every $x \in \{0,1\}^*$,

$$x \in L \Leftrightarrow \forall u \in \{0,1\}^{p(|x|)} \text{ s.t. } M(x,u) = 1$$

coNP Complete

TAUTOLOGY is $\mathbf{coNP}$-complete

In classical logic, tautologies are true statements. The following language is $\mathbf{coNP}$-complete:

TAUTOLOGY $= \{\varphi : \ \varphi$ – Boolean formula that is satisfied by every assignment$\}$.

It is clearly in $\mathbf{coNP}$ and so all we have to show is that for every $L \in \mathbf{coNP}$, $L \leq_p$ TAUTOLOGY. But this is easy: just modify the Cook-Levin reduction from $\overline{L}$ (which is in $\mathbf{NP}$) to SAT. For every input $x \in \{0,1\}^*$ that reduction produces a formula φ_x that is satisfiable iff $x \in \overline{L}$. Now consider the formula $\neg\varphi_x$. It is in TAUTOLOGY iff $x \in L$, and this completes the description of the reduction.

coNP Complete

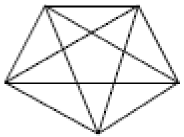
coNP-complete problems

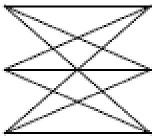
- Complements of NP-complete problems
- UNSAT: Given Boolean formula, is it unsatisfiable?
- TAUTOLOGY (VALIDITY): Given Boolean formula, is it a tautology (valid), i.e. satisfied by all truth assignments?
- NONHAMILTONICITY: Given a (undirected or directed) graph, is it nonHamiltonian?
- NON 3-COLORABILITY: Given an undirected graph, is it the case that it has no 3-coloring?
- NODE COVER LOWER BOUND: Given graph G and number k, does every node cover of G have $\geq$k nodes?
- INDEPENDENT SET UPPER BOUND: Given a graph G and number k, does every independent set of G have $\leq$k nodes?

NP and coNP

NP∩coNP

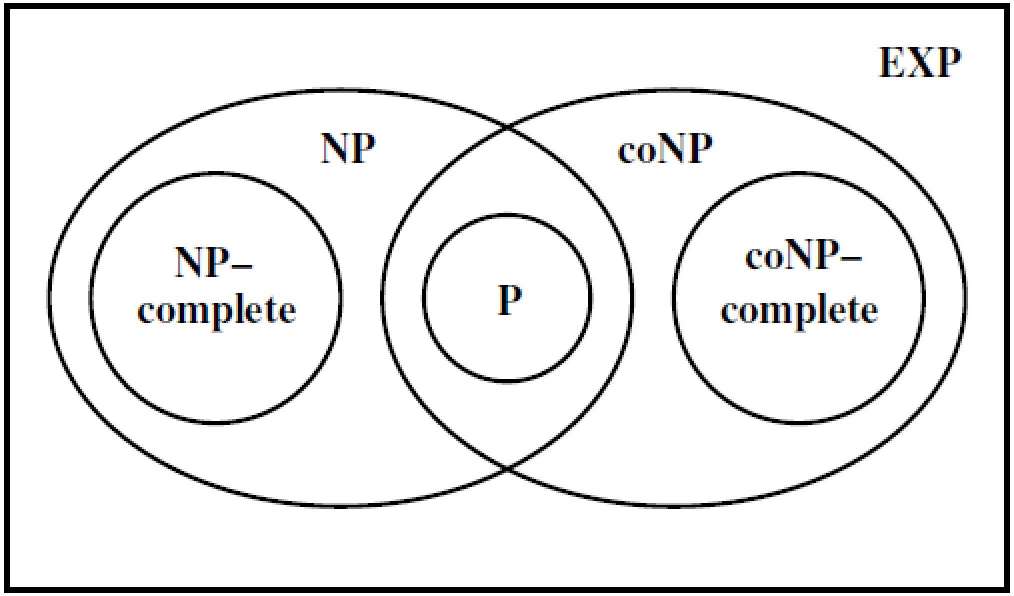
- Short, easy to check certificates both for the Yes and the No instances
- Examples:
- Graph Bipartiteness:
  - bipartite ⇔ nodes can be partitioned into two sets V1, V2 so that all edges connect a node in V1 with a node in V2
  - nonbipartite ⇔ there is an odd length cycle
- Graph Planarity
  - planar ⇔ can draw on the plane so that no edges intersect
  - nonplanar ⇔ contains a homeomorph of K_5 or K_{33} (Kuratowski's theorem)

K_5 

K_{33} 

- These particular properties happen to be in fact in P

P, NP, coNP



- NP is closed under union, intersection
- coNP is also closed under union, intersection

- NP (and coNP) closed under complement iff NP=coNP
- conjectured not

NP-naming convention

- NP-complete - means problems that are 'complete' in NP, i.e. the most difficult to solve in NP
- NP-hard - stands for 'at least' as hard as NP (but not necessarily in NP);
- NP-easy - stands for 'at most' as hard as NP (but not necessarily in NP);
- NP-equivalent - means equally difficult as NP, (but not necessarily in NP);

Decision Vs. Optimization

- Decision problem: a question that has two possible answers yes or no. The question is about some input.

- Optimization problem: find a solution that maximizes or minimizes some objective function

Decision Vs. Optimization

Decision problem:

- Given a graph G and a set of vertices K, is K a clique?
- Given a graph G and a set of edges M, is M a spanning tree?
- Given a set of axioms (boolean expressions) and an expression, is the expression provable under the axioms?

Decision Vs. Optimization

- Optimization problems are not stated as "yes/no' questions.
- An optimization problem can be transformed to a decision problem using a bound on the solution
- Example:
  - TSP Optimization: Find the shortest path that visits all cities
  - TSP Decision: Is there a path of length smaller than B?

Max2Sat is NP-Complete

Max-2-SAT

Instance:

- a 2-CNF formula φ

Maximization Problem:

- Find the *maximum #* of clauses satisfied by an assignment to φ

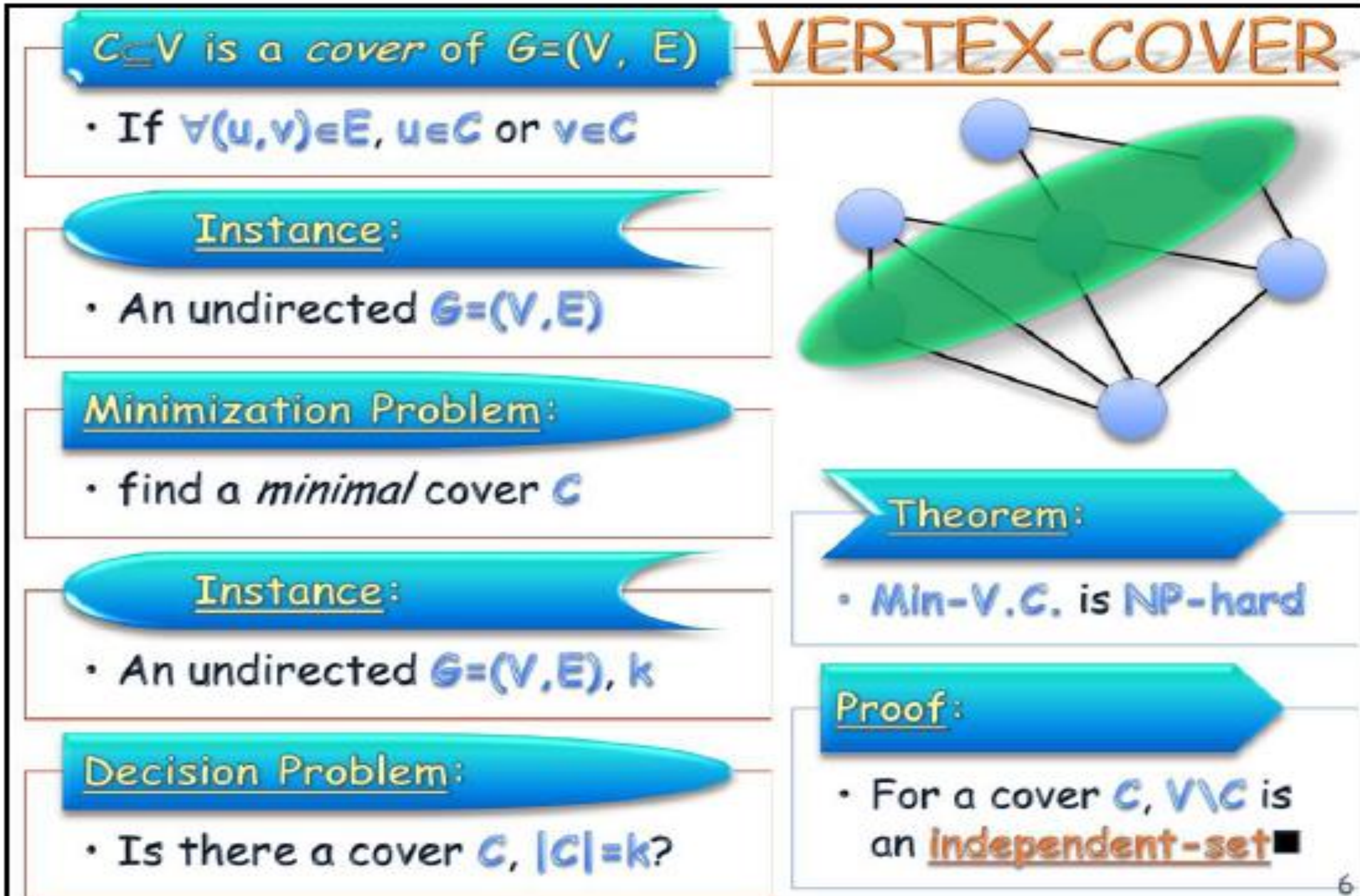
Instance (decis. ver.):

- a 2-CNF formula φ and a *threshold* K

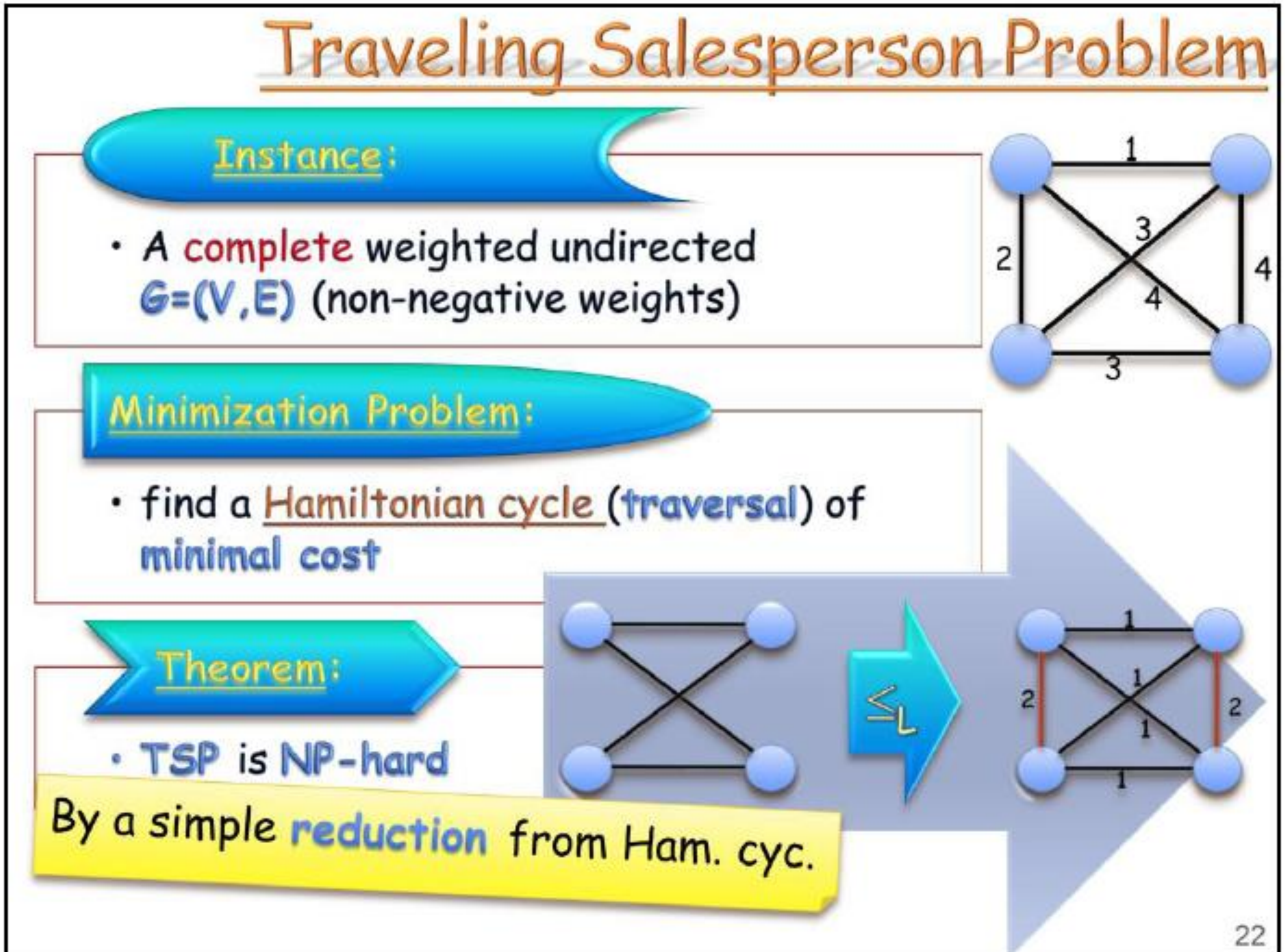
Decision Problem:

- Is there an assign. satisfying $\geq K$ clauses of φ?

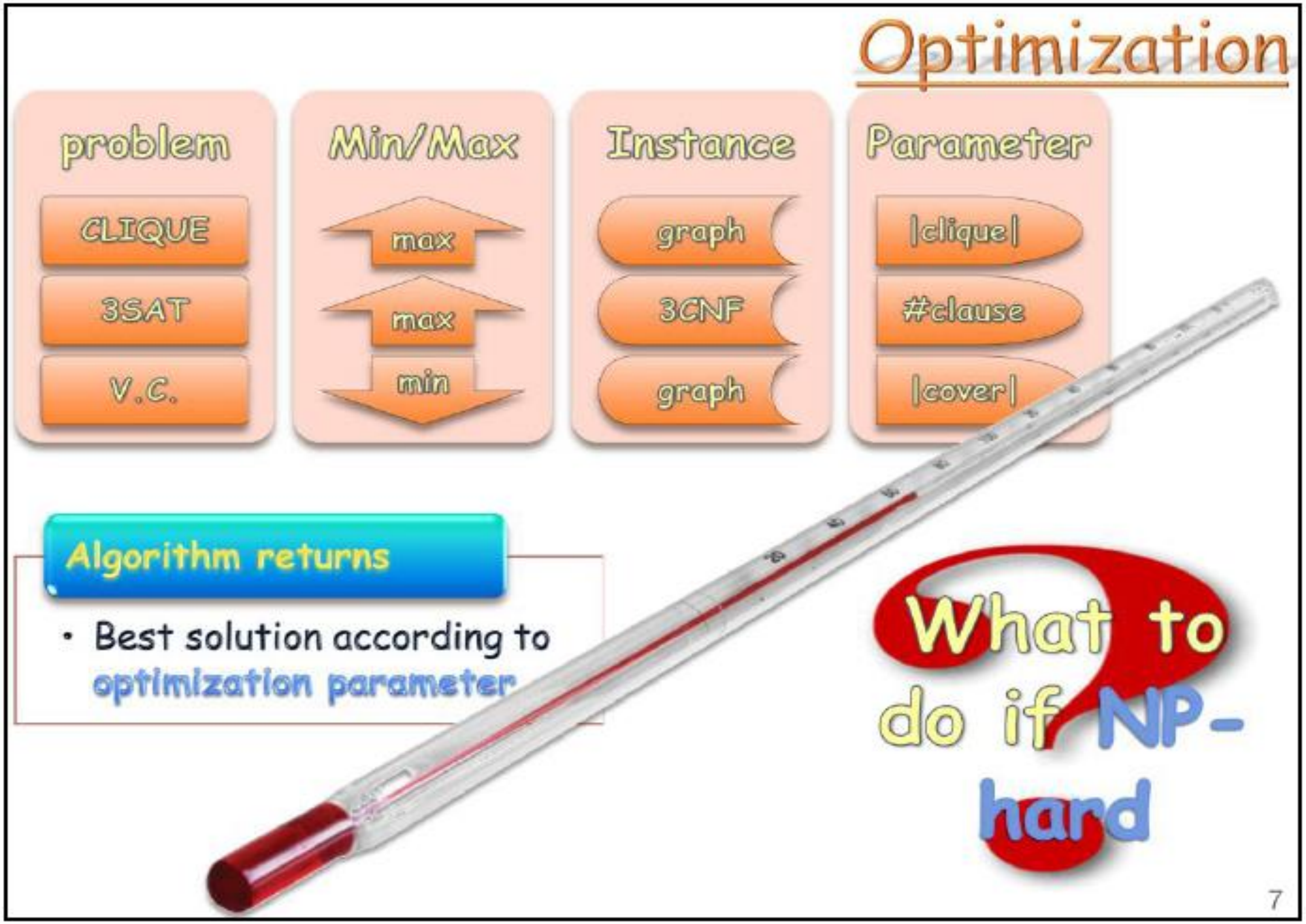
NP-hard problems: example



NP-hard problems: example



NP-hard problems: example



NP-hard problems: example

- **Halting Problem** is NP-hard decision problem, but it is not NP-complete.

- For this let us construct an algorithm A whose input is a prepositional formula X.
- Suppose X has n variables. Algorithm A tries out all 2^n possible truth assignments and verifies if X is satisfiable.
- If it is satisfied then A stops. If X is not satisfiable, then A enters an infinite loop. Hence A halts on input iff X is satisfiable.
- If we had a polynomial time algorithm for the halting problem, then we could solve the satisfiability problem in polynomial time using A and X as input to the algorithm for the halting problem .

Complexity Classes EXP, NEXP

EXP and NEXP

The following two classes are exponential time analogues of P and NP.

Def:
- $\mathrm{EXP} = \cup_{c \geq 0} \mathrm{DTIME}(2^{n^c})$.
- $\mathrm{NEXP} = \cup_{c \geq 0} \mathrm{NTIME}(2^{n^c})$.

Because every problem in NP can be solved in exponential time by a brute force search for the certificate, $\mathrm{P} \subseteq \mathrm{NP} \subseteq \mathrm{EXP} \subseteq \mathrm{NEXP}$.

Is there any point to studying classes involving exponential running times?

The following simple result may be a partial answer.

Complexity Class EXP

- Generalized chess is the game of chess played on an *n*-by-*n* board, with 2*n* pieces on each side.

- For many generalized games which may last for a number of moves exponential in the size of the board, the problem of determining if there is a win for the first player in a given position is EXPTIME-complete.

- SUCCINT representation of P problems.

Complexity Class NEXP

- **SUCCINCT HAMILTON PATH:**

  A Boolean circuit with ***2n*** inputs and one output represents a graph on 2^n vertices. To determine if there is an edge between vertices ***i*** and ***j***, encode ***i*** and ***j*** in ***n*** bits each, and feed their concatenation to the circuit: there is an edge between these vertices iff the output of the circuit is true.

$$C : \{0,1\}^n \times \{0,1\}^n \mapsto \{0,1\}$$

$$G = \left(\{0,1\}^n, \{(u,v) : C(u,v) = 1\}\right)$$

- Given such a circuit, is there a Hamilton path in the graph represented by the circuit?

$$L = \{C : C \text{ describes a graph with a Hamiltonian cycle}\}.$$

- For some NP-complete problems, there's a SUCCINCT variant that's NEXP-complete. E.g., SUCCINCT 3SAT, SUCCINCT KNAPSACK, etc.

Complexity Class NEXP

Definition 2.1 TILING

Problem Parameters: *A set of tiles* $T = \{t_1, \ldots, t_m\}$. *A set of horizontal constraints* $H \subseteq T \times T$ *such that if* t_i *is placed to the left of* t_j, *then it must be the case that* $(t_i, t_j) \in H$. *A set of vertical constraints* $V \subseteq T \times T$ *such that if* t_i *is placed below* t_j, *then it must be the case that* $(t_i, t_j) \in V$. *A designated tile* t_1 *that must be placed in the four corners of the grid.*

Problem Input: *Integer* N, *specified in binary.*

Output: *Determine whether there is a valid tiling of an* $N \times N$ *grid.*

Theorem 2.2 *TILING is* NEXP-*complete.*

Satisfyability of true boolean quantified formulas is NEXP.

3.1 EXP and NEXP-complete problems

One general method is to find *succinct* versions of **P**, **NP**-complete problems. But what is a succinct representation of a graph?

Definition 3 *A* succinct representation of a graph *is a circuit* $C : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ *defining* $G_C = (V, E)$, $V = \{0,1\}^n$, $E = \{(u,v) : C(u,v) = 1\}$

We then have a potentially succinct representation of the graph, since it can represent something exponentially larger than its description length (the circuit).

This yields succinct versions of familiar graph problems.

Definition 4 Succinct Hamiltonian Path*:* SHP $= \{C : G_C$ *has a Hamiltonian path*$\}$.

Proposition 5 SHP $\in$ **NEXP**

Theorem 6 Succinct Circuit Value *is* **EXP**-*complete. Also,* Succinct Circuit SAT *is* **NEXP**- *complete.*

Complexity Classes EXP, NEXP

If $\mathrm{EXP} \neq \mathrm{NEXP}$ then $\mathrm{P} \neq \mathrm{NP}$

We prove the contrapositive: $\mathrm{P} = \mathrm{NP}$ implies $\mathrm{EXP} = \mathrm{NEXP}$.

Suppose $L \in \mathrm{NTIME}(2^{n^c})$ and NDTM M decides it. We claim that then the language

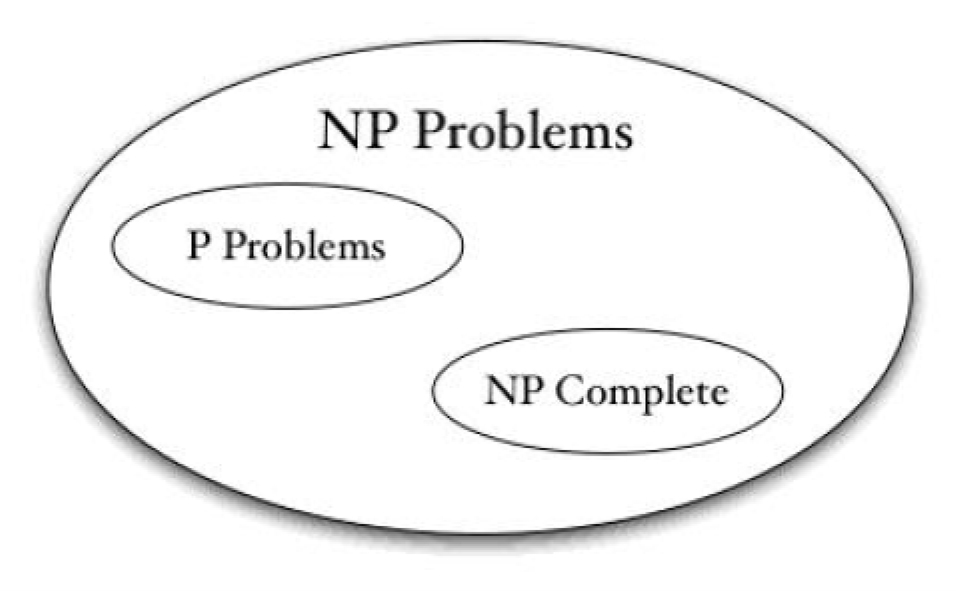
$$L_{\text{pad}} = \{\langle x, 1^{2^{|x|^c}}\rangle : x \in L\}$$

is in NP. Here is an NDTM for L_{pad}:

- given y, first check if there is a string z such that $y = \langle z, 1^{2^{|z|^c}}\rangle$. If not, output REJECT.
- If y is of this form, then run M on z for $2^{|z|^c}$ steps and output its answer.

Clearly, the running time is polynomial in $|\ y\ |$, and hence $L_{\text{pad}} \in \mathrm{NP}$. Hence if $\mathrm{P} = \mathrm{NP}$ then L_{pad} is in P. But if L_{pad} is in P then L is in EXP: to determine whether an input x is in L, we just pad the input and decide whether it is in L_{pad} using the polynomial-time machine for L_{pad}.

Complexity Classes P and NP



Source: Wikipedia (Complexity Classes P and NP)

Your Chance to be Famous

The question of whether P is the same set as NP is the most important open question in theoretical computer science. There is even a $1,000,000 prize for solving it.

Source: Wikipedia (Clay Mathematics Insitute)