NP

From deterministic to nondeterministic

Everything defined so far is based on deterministic TMs

- What about *nondeterministic* TMs?
- No difference on computability, but huge difference on running time

Accept/Reject are no longer symmetric!

- To Accept, one accepting run suffices to “certify”
- To Reject, all runs must reject!

The class NP

Definition (class NTIME): Let $T : \mathbb{N} \to \mathbb{N}$ be some function. A language L is in $\mathbf{NTIME}(T(n))$ iff there is a nondeterministic Turing machine (NTDM) M that runs in time $O(T(n))$ and decides L.

- $x \in L$ if and only if at least one run of M accepts x.

Definition: $\mathbf{NP} = \bigcup_{c \geq 1} \mathbf{NTIME}(n^c)$

Definition: $\mathbf{coNP} = \{A \mid \bar{A} \in \mathbf{NP}\}$

- $x \notin L$ if and only if at least one run of M rejects x.

Proposition:

- $\mathbf{P} \subseteq \mathbf{NP} \cap \mathbf{coNP}$

Open problem:

- $\mathbf{NP} = \mathbf{coNP}$?

An alternative characterization of NP

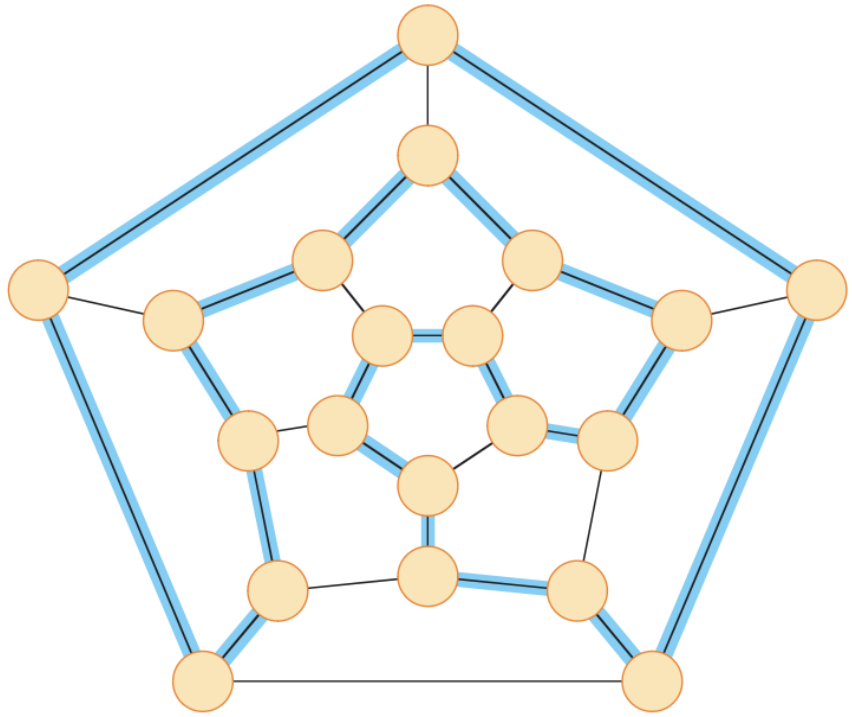
Definition: A language A is *polynomially verifiable* if there is a $k \in \mathbb{N}$ and a deterministic Turing machine V such that $A = \{w \mid \exists p.\ V \text{ accepts } \langle w, p \rangle\}$ and V takes at most $|w|^k$ steps on input $\langle w, p \rangle$, i.e., V running time is *independent* of the length of p.

We call p a *certificate* for w (w.r.t. A and V).

Example

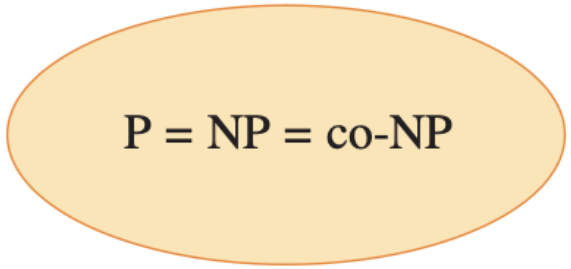
Graph Hamiltonicity:

- Input: an undirected graph $G = (V, E)$
- Certificate: a permutation of V
- Verification: in $\Theta(n)$ time, check that the permutation represents a Hamiltonian cycle

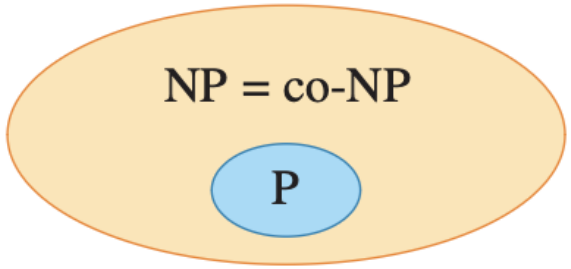


P = NP ?

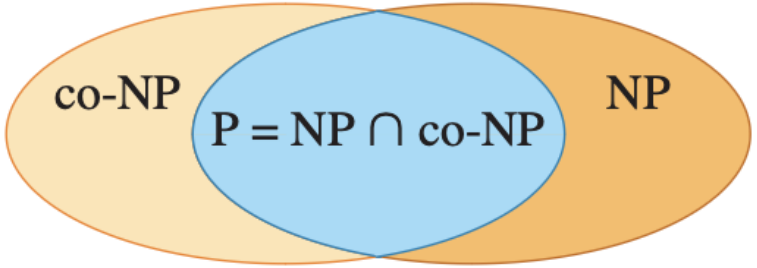
Two open problems, four possible cases



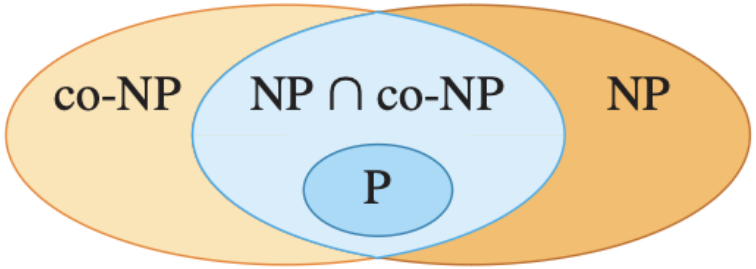
(a)



(b)



(c)



(d)

Philosophical importance of NP

What is the power of nondeterminism?

- Answer is clear for finite automata, but not for Turing machines

Can exhaustive search be avoided?

- Grading quizzes vs. taking quizzes?
- Multiplication is far easier than factorization
- “Appreciating a Beethoven sonata is far easier than composing the sonata” (Arora and Barak)