Def 7.12: The class $P$ consists of languages that are decidable in polynomial time, i.e.,

$$P = \bigcup_k \text{TIME}(n^k)$$

Example:

$$\text{PATH} = \{ <G,s,t> \mid G \text{ is a digraph that has a path from } s \text{ to } t \}$$

Other examples:
- Given a list of numbers, is the list sorted?
- etc.

solve e.g. via BFS or DFS \( \Rightarrow \) take \( O(n+m) \)
(or Dijkstra with weights 1)

where
\( n = \# \text{nodes} \)
\( m = \# \text{edges} \)

\( \Rightarrow \) BFS/DFS is poly-time \( \text{PATH} \in P \)
The Class P

Def 7.12: The class P consists of languages that are decidable in polynomial time, i.e.,

\[ P = \bigcup_k \text{TIME}(n^k) \]

Example:

RELPRIME = \{ <x,y> \mid x \text{ and } y \text{ are relatively primes} \}

- If \( x, y \) have, e.g., \( k \) digits each, we need to try \( 10^k \) possible divisors. Not a poly-time approach.
- Other approach: Euclidean algo for finding the gcd (greatest common divisor) \( \rightarrow \) takes time \( O(k) \).

Thus: RELPRIME \( \in P \)
Def 7.12: The class $P$ consists of languages that are decidable in polynomial time, i.e.,

$$P = \bigcup_k \text{TIME}(n^k)$$

Example:

PRIME = \{ <x> | x is a prime \}

idea: divide $x$ by all numbers $2, 3, \ldots, \sqrt{x}$ and see if any of them divides $x$

$\Rightarrow$ if yes, $x$ is not a prime

$\Rightarrow$ if no, $x$ is a prime

problem: exponential time

Miller-Rabin - a fast randomized test

Agrawal-Saxena-Kayal - showed that PRIME $\in$ P [2002]
The Class \( \text{NP} \)

Example:

\[ \text{HAMPATH} = \{ <G,s,t> \mid G \text{ is digraphs with Hamiltonian path from } s \text{ to } t \} \]

\( P \): find the solution in (det) poly-time
\( \text{NP} \): given a "solution" (a "guess")
- verify its correctness in poly-time

\( \text{poly-time verification} \)
- verify that
  - first vertex is } s
  - the last vertex is } t
  - every other vertex visited exactly once
  - there is an edge between every pair of consecutive vertices

\( \text{nondeterministically "guess" a sequence of vertices} \)

\( \text{is HAMPATH } \in \text{ NP? } \quad \text{YES} \)
The Class NP

Example:

\[ \text{COMPOSITES} = \{ x \mid x = pq, \text{ for some } p, q > 1 \} \]

\[ \in \text{NP} \quad \text{YES} \]

\textbf{sketch:} \quad \text{first nondeterministically "guess" } p \text{ and } q \quad \text{verify:}
\[
\text{that } x = p \cdot q, \text{ if yes, then } x \text{ is a composite}
\]
Def 7.18: A **verifier** for a language $A$ is an algorithm $A$, where

$$A = \{ w \mid V \text{ accepts } <w, c> \text{ for some string } c \}$$

**Polynomial-time verifier** runs in (deterministic) time polynomial in the length of $w$. The string $c$ is called the **certificate**, or **proof**, of the membership in $A$. 
**Def 7.18**: A **verifier** for a language $A$ is an algorithm $A$, where

$$A = \{ w \mid V \text{ accepts } \langle w, c \rangle \text{ for some string } c \}$$

**Polynomial-time verifier** runs in (deterministic) time polynomial in the length of $w$. The string $c$ is called the **certificate**, or **proof**, of the membership in $A$.

**Def 7.19**: $\mathbf{NP}$ is the class of languages that have polynomial time verifiers.
**The Class NP**

**Thm 7.20:** A language is in NP iff it is decided by some nondeterministic polynomial-time TM.

**Def 7.21:** \( \text{NTIME}(t(n)) = \{ L \mid L \text{ is a language decided by a } O(t(n))-\text{time nondeterministic TM} \} \).

Thus, \( \text{NP} = \bigcup_k \text{NTIME}(n^k) \)
The Class NP

Thm 7.20: A language is in NP iff it is decided by some nondeterministic polynomial-time TM.

Def 7.21: \( \text{NTIME}(t(n)) = \{ L \mid L \text{ is a language decided by a } O(t(n))-\text{time nondeterministic TM} \} \).

Thus, \( \text{NP} = \bigcup_k \text{NTIME}(n^k) \)
The Class NP

Example:

\[
\text{CLIQUE} = \{ <G,k> \mid G \text{ is undirected graph with a } k\text{-clique} \}
\]
Example:

\[ \text{SAT} = \{ \langle \phi \rangle \mid \phi \text{ is a satisfiable Boolean formula} \} \]
The Class NP

Wrapping up:
- P - exists polynomial-time algorithm
- NP - exists polynomial-time verifier

BIG open problem:

Is P = NP ???

Note: also exists a class coNP, the class of complements of problems in NP (e.g. CLIQUE^c, “is every clique of a given graph of different size than k?”). We do not know if NP = coNP.