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:

PATH = \{ <G,s,t> | G is a digraph that has a path from s to t \}
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> | x and y are relatively primes \}

```plaintext
for i=2 to min{x,y}:
    if i divides x and i divides y:
        return False (not rel. prime)
return True
```

\[ \text{Correct algo}\]
\[ \text{run.time: } O(\min{x,y}) \]

\[ \text{Not a poly-time algo}\]
\[ x = 10^{1000}\]
\[ y = 10^{1000} + 1 \]
\[ \text{size of the input: } \sim \log x \text{ digits} + \log y \text{ digits} \]
\[ \text{suppose } x \neq y: \ k = \# \text{digits of } x \]
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} \}

\[ \text{in } P: \quad \text{Euclid's Algorithm for } \gcd \]

\[ \rightarrow \text{ takes } O(\log \max\{x,y\}) \text{ steps} \]

\[ \text{polynomial in } k, \text{ the length of the input, } k=\text{length} \]

\[ \rightarrow \text{ about } O(k) \text{ steps} \]

k \sim \log x + \log y
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:

\[
\text{PRIME} = \{ \langle x \rangle \mid x \text{ is a prime} \}
\]

```plaintext
for i = 2 to \( \sqrt{x} \):
  if i divides x:
    return false
return true
```
The Class NP

Example:

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

- Hamiltonian
- "framework"
- "guess": a permutation of \( n \) vertices (for HAMPATH)
- verify (deterministic):
  - start in \( s \)
  - end in \( t \)
  - every pair of consecutive vertices is connected by an edge

- goes through every single vertex exactly once
- poly \# steps
- poly \# steps

[Section 7.3]
Example:

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

in \( \mathcal{P} \) because of the AKS-primality testing.

But we know \( \in \mathcal{NP} \) before.

\[
\begin{align*}
\text{guess:} & \quad p, q \quad \text{s.t.} \quad 1 < p, q < x \\
\text{verify:} & \quad x = p \cdot q
\end{align*}
\]
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 $A$. 
The Class NP

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 7.19: 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: 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: NTIME(t(n)) = { L | L is a language decided by a $O(t(n))$-time nondeterministic TM }.

Thus, $NP = \bigcup_k NTIME(n^k)$
The Class NP

Example:

$\text{CLIQUE} = \{ <G,k> \mid G \text{ is undirected graph with a } k\text{-clique} \}$

<table>
<thead>
<tr>
<th>$k$</th>
<th>In NP:</th>
<th>Clique:</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>guess: a set of $k$ vertices</td>
<td>every pair connected by an edge</td>
</tr>
<tr>
<td>4</td>
<td>verify: edge between every pair</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>
The Class NP

Example:

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

Boolean formula in the CNF (conjunctive normal form)

\[ \phi = (x_1 \lor x_2 \lor \neg x_3) \land (\neg x_1 \lor x_2 \lor x_3) \land (\neg x_1 \lor \neg x_2) \]

YES, satisfiable because:

- \(x_1 = \text{F}\)
- \(x_2 = \text{T}\)
- \(x_3 = ?\)

in NP

guess:

T/F for each variable

verify:

statement is true
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.