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:

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

e.g. BFS or DFS from \( s \), checking if can hit \( t \\
\text{running time: } O(n+m) \\
\text{or even a more crude upperbound } O(n^2) \text{ will show that} \\
\text{(on a TM running a bit higher, still 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:**

$\notin P$

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

**Idea 1:**

for $i = 2$ to $\min(x,y)$:

if $i$ divides $x$ and $y$, then not rel-prime

return rel-prime

Run time: $O(\min(x,y))$ iterations

= $O(10^n)$ iterations

 exponential time algo

**Idea 2:** Euclid to compute $\gcd(x,y)$; if 1 then rel-prime

(Euclid takes $O(\log x \log y)$ steps)

= $O(n)$ poly time
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:

PRIME = \{ <x> \mid x \text{ is a prime} \} \quad \in P \quad \text{AKS primality checking (2002)}

for $i = 2$ to $\sqrt{x}$:
  if $i$ divides $x$, then not prime.
return prime

Not pseudo-time:
$$O(\sqrt[10]{n}) = O(\sqrt[100]{n})$$
when $n = \log x$ digits.
The Class NP

Example:

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

Example:

$COMPOSITES = \{ \langle x \rangle \mid x = pq, \text{ for some } p, q > 1 \}$

in NP

1. guess:
   integer $p > 1$ and $p < x$

2. verify:
   does $p$ divide $x$

(Btw in $\mathbb{P}$
bec. PRIME $\Rightarrow$)


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 | L is a language decided by a O(t(n))-time nondeterministic TM \}.

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

Example:

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

1. guess: \( k \) vertices
2. verify:
   - every pair connected by an edge
   - no duplicate vertices

in \( \text{NP} \):

\( k = 4 \):

- yes

\( k = 5 \):

- no (maybe)

we do not know if \( \epsilon \) \( \in \) \( \text{P} \)

A set of \( k \) vertices mutually connected by edges.
Example:

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

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

Is there a true/false assignment to the variables so that \( \phi \) is true?

In NP:

1. guess:
   - boolean assignment to the variables

2. verify:
   - is \( \phi \) with the assignment true?

\[ x_1 = F \]
\[ x_2 = T \]
\[ x_3 = F \]

Then \( \phi = T \)

not know if \( \in P \)
The Class NP

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

BIG open problem:

Is \( P = \text{NP} \) ???

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