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:

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

- do a BFS from s, see if find t
- notice that BFS runs in time \( O(|V| + |E|) \)
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} \}

Idea 1: for $z = 2$ to $x$:

- if $z$ divides both $x$ and $y$, return NO //not relatively prime

- return YES

notice that $|x| = \log_{10} x + 1$

Idea 2: gcd - Euclidean algo \rightarrow in $P$ \checkmark

bec. takes time $O(\log x + \log y)$
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:

$\text{PRIME} = \{ <x> \mid x \text{ is a prime} \}$

1. for $i = 2$ to $\sqrt{x}$: check if $i$ divides $x$  \(\leq\) exponential time
2. randomized poly-time by Miller-Rabin  \(\leq\) does not show it is in $P$ (used in practice \(\rightarrow\) fast)
3. recent: Agrawal, Kayal, Saxena  \(\rightarrow\) in $P$ (involved number theory)
The Class NP

Example:

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

i.e. goes through all other vertices (exactly once)

NP \rightarrow the class of languages for which there exists a poly-time nondet. TM

HAMPATH \in NP \rightarrow bec. the TM can nondeterministically go through all paths, if one gets to s, check if it contains all vertices and if yes, accept

(time proportional to the length of the path = poly-time)

overall \( O(n^2) \) steps

also needs \( O(n^2) \) steps

same as:

1. nondeterministically guess (generate) a sequence of vertices
2. deterministically verify that sequence contains all vertices, each once adjacent vertices connected by an edge starts in s, ends in t

needs \( O(n^2 \text{ vertices}) \) steps
The Class NP

Example:

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

**nondet. poly-time algo:**

1. nondet. guess \( p \) and \( q \)
2. det. verify:
   - \( p \) is a prime
   - \( q \) is a prime

\[ \text{and } p \cdot q = x \text{ is poly-time} \]

**overall: poly-time verifier**

Another idea:

for \( i = 2 \) to \( \sqrt{x} \):

- check if \( i \) is a prime; if yes, check if \( i \) divides \( x \) and gets a prime;
- if yes, return "YES, a composite"
- return "NO, not a multiple of 2 primes"

exponential time, see the \text{PRIME} and \text{RELPRIME} slides
Def 7.18: A **verifier** for a language $A$ is an algorithm $A$, where

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

**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 $A$. 

NP:

1. non-det. guess of the solution (aka certificate/proof)
2. det. poly-time verifier of the solution
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: $\text{NTIME}(t(n)) = \{ L \mid L \text{ is a language decided by a } O(t(n))\text{-time nondeterministic TM} \}$.

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

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

Example:

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

\[
\begin{align*}
G: & \\
\text{k = 3} & \text{YES} \\
\text{k = 4} & \text{YES} \\
\text{k = 5} & \text{NO}
\end{align*}
\]

nondet. poly-time algo:

1. nondet. guess \( k \) vertices
2. det. verify if all distinct and every pair connected by an edge

\[
\text{needs } O(k) \quad \text{needs } O(k^2) \quad \text{overall: poly-time}
\]
The Class NP

Example:

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

\[
\text{e.g. } \phi = (x_1 \lor x_2 \lor x_3) \land (x_2 \lor x_3) \land (x_1 \lor x_2)
\]

\[
\text{e.g. } x_1 = \text{true} \quad x_2 = \text{true} \quad x_3 = ?
\]

Then \( \phi = \text{true} \)

\[
\text{in NP:}
\]

1. guess (non-det.) a T/F assignment to each variable
2. det. verify that: \( \phi \) is true, given this assignment
The Class NP

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

BIG open problem:

Is \( P = NP \) ???

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