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> \mid G \text{ is a digraph that has a path from } s \text{ 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 \}

- Compute gcd(x,y)
  - If different than 1, not rel. prime
  - Euclid's algo: \( O(\log(x+y)) \)
    - Linear in \( n \)
    - \( O(n) \)

- The length of the input is at most:
  - \( n = \lceil \log x + \log y + 3 \rceil \)
  - need about \( \log(x) \) digits (if decimal notation, log base 10)
  - e.g. \( x = 10^{1003} \) we need 1004 digits

- \( \text{i.e. don't have a divisor} > 1 \)

- \( \text{b/c the larger number becomes halved or less every 2 steps} \)
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:

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

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

  return prime

runtime: $O(\sqrt{x}) = O((\log n)^{1/2}) = O\left(\frac{\log n}{\log \log n}\right)$

(size of input:

$n = \lceil \log_{10} x + 1 \rceil$

$X \approx 10^n$

feels like not in $P$)
The Class NP

Example:

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

Example:

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

NP:

1. guess \( p > 1, p < \sqrt{x} \)
2. verify whether \( p \) divides \( x \)

Notice: nondet. polytime

Btw (Tim):

can run primality checking & return the opposite answer
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 $A$. 
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, \[ \text{NP} = \bigcup_k \text{NTIME}(n^k) \]
Example:

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

don't know it in P

NP:
1. guess k vertices
2. verify:
   - all connected
   - all vertices different
The Class NP

Example:

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

\[ \phi = (x_1 \lor x_2) \land (x_3 \lor x_2) \land (x_4 \lor x_1) \land (x_3 \lor x_2 \lor x_1) \]

Can assign True/False values to \( x_1, …, x_4 \) s.t. \( \phi \) is true?

NP:
1. guess T/F assignment for all variables
2. verify if \( \phi \) is true

\[ \text{don't know if in P} \]
The Class NP

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

BIG open problem:

\[
\text{Is } P = NP \text{ ???}
\]

**Note**: also exists a class 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 \( NP = \text{coNP} \).