A more formal approach using languages

The language of a decision problem is the set of all inputs for which the answer is YES, encoded as finite strings.

\[ \text{HAMCYCLE} = \{ G : G \text{ is a graph that contains a Hamilton cycle} \} \]

\[ \text{TSP} = \{ (C,K) : C \text{ is an integer nxn matrix,} \]
\[ K \text{ is an integer,} \]
\[ \exists \text{ permutation } \pi \text{ of } 1, \ldots, n \text{ such that} \]
\[ \sum_{i=1}^{n-1} C_{\pi_i \pi_{i+1}} + C_{\pi_n \pi_1} \leq K \} \]
\text{SUBSETSUM} = \{(S, t) : S \text{ is a set of integers, } t \text{ is an integer, } \\
\exists S \subseteq S : \sum_{x \in S} x = t \}\}

\text{CLIQUE} = \{(G, K) : G \text{ is a graph, } \\
K \text{ is an integer, } \\
G \text{ contains a clique with } K \text{ vertices } \}$
Definition of the class $\mathcal{P}$:

The language $L$ is in $\mathcal{P}$, if the following is true:

There exists an algorithm $A$ and a constant $c \geq 1$, such that for any input $x$:

* if $x \in L$, then $A(x)$ returns YES
* if $x \notin L$, then $A(x)$ returns NO
* the running time of $A(x)$ is $O(n^c)$, where $n$ is the length of $x$. 
Definition of the class NP:
The language $L$ is in NP, if the following is true:

There exists an algorithm $V$ and a constant $c \geq 1$, verification algorithm, takes 2 input parameters such that for any input $x$:

$x \in L \iff$ there exists a certificate $y$ such that

$|y| = O(|x|^c)$,

$V(x,y)$ returns YES, and

the running time of $V(x,y)$ is polynomial in the length of $x$.

NP stands for non-deterministic polynomial time.
We show that
\[ \text{HAMCYCLE} = \{ G : G \text{ is a graph that has a Hamilton cycle} \} \]
is in NP:

Verification algorithm V takes as input

* graph G
* certificate \( v_1, \ldots, v_n \)

**Step 1:** check if \( \{ v_1, \ldots, v_n \} = \text{vertex set of } G \).

**Step 2:** check if \( |\{ v_1, \ldots, v_n \}| = n \).

**Step 3:** check if \( \{ v_1, v_2 \}, \{ v_2, v_3 \}, \ldots, \{ v_{n-1}, v_n \}, \{ v_n, v_1 \} \)
are edges in G.

**Step 4:** if Steps 1-3 were successful, return YES; otherwise, return NO.
$G$ is in HAMCYCLE

$\Leftrightarrow \exists$ permutation $v_1, \ldots, v_n$ of $G$'s vertex set such that

$\{v_1, v_2\}, \{v_2, v_3\}, \ldots, \{v_{n-1}, v_n\}, \{v_n, v_1\}$

are edges in $G$

$\Leftrightarrow \exists$ certificate $(v_1, \ldots, v_n)$ such that

$V(G, (v_1, \ldots, v_n))$ returns YES.

The length of the certificate

$= \# \text{ vertices in } G = O(\text{ size of } G)$.

Running time of $V$: $O((\text{ size of } G)^2)$. 
Claim: $P \subseteq NP$.

Proof: Let $L$ be an arbitrary language in $P$.
By definition, there is an algorithm $A$ such that for any input $x$:
- $x \in L \implies A(x)$ returns YES
- running time of $A(x)$ is polynomial in the length of $x$.

We have to show that $L$ is in $NP$.
The verification algorithm $V$ takes as input
- the input $x$ for $L$,
- certificate $y$.
$V(x, y)$ does the following: run $A(x)$.

(thus, $V$ ignores $y$)
\( x \in L \iff A(x) \text{ returns YES} \)

\( \iff \forall (x, \text{ empty string } y) \text{ returns YES} \)

\( \iff \exists \text{ certificate } y \text{ such that } \)

\[ \text{length of } y = 0 = \text{ polynomial in the length of } x, \]

\( V(x, y) \text{ returns YES} \)

and running time of \( V(x, y) = \)

running time of \( A(x) = \text{ polynomial in the length of } x. \)

Therefore, \( L \) is in \( NP \).

\( \square \)

**Big Question**: Is \( P = NP \) or \( P \neq NP \)?

Most people believe that \( P \neq NP \).
If we want to prove that $P \neq NP$, then we have to show that there exists a language $L$ such that:

* $L \in NP$
* $L \not\in P$

Such an $L$ must be "difficult".

$\Rightarrow$ Look at the "most difficult" problems in $NP$.

What does this mean?

how to compare problems by their difficulty?

$\Rightarrow$ reductions
Definition of reduction

Let \( L \) and \( L' \) be languages.

\[
L \leq_P L' \quad \text{if } L \text{ is polynomial-time reducible to } L',
\]

\( L' \) is at least as difficult as \( L \)

if the following is true:

There exists a function \( f \) such that

1. \( f \) maps inputs to \( L \) to inputs to \( L' \)
2. for every input \( x \) to \( L \):
   \[
   x \in L \iff f(x) \in L'
   \]
3. for every input \( x \) to \( L \):
   \( f(x) \) can be computed in time that is polynomial in the length of \( x \).