A more formal approach using languages

The language of a decision problem is the set of all $\underbrace{\text{inputs}}_{\text{encoded as finite strings}}$ for which the answer is YES.

HAMCYCLE = { G : G is a graph that contains a Hamilton cycle }

TSP = { (C, K) : C is an integer n x n matrix,
K is an integer,
$\exists$ permutation π of $1, \dots, n$ such that

$$\sum_{i=1}^{n-1} C_{\pi_i, \pi_{i+1}} + C_{\pi_n, \pi_1} \leq K \}$$

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 <u>non-deterministic</u> polynomial time.

We show that

HAMCYCLE = $\{G : G$ is a graph that has a Hamilton cycle$\}$

is in NP:

Verification algorithm V takes as input

* graph G
* certificate $v_1, \dots, v_n$

<u>Step 1</u>: check if $\{v_1, \dots, v_n\}$ = vertex set of G.

<u>Step 2</u>: check if $|\{v_1, \dots, v_n\}| = n$.

<u>Step 3</u>: check if $\{v_1, v_2\}, \{v_2, v_3\}, \dots, \{v_{n-1}, v_n\}, \{v_n, v_1\}$ are edges in G.

<u>Step 4</u>: if Steps 1–3 were successful, return YES; otherwise, return NO.

G is in HAMCYCLE

$\Longleftrightarrow$ $\exists$ 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

$\Longleftrightarrow$ $\exists$ certificate $(v_1, \ldots, v_n)$ such that $V(G, (v_1, \ldots, v_n))$ returns YES.

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

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

Claim: $\mathcal{P} \subseteq \mathcal{NP}$.

Proof: Let L be an arbitrary language in $\mathcal{P}$.

By definition, there is an algorithm A such that for any input x:

* $x \in L \Longleftrightarrow A(x)$ returns YES
* running time of $A(x)$ is polynomial in the length of x.

We have to show that L is in $\mathcal{NP}$.

The verification algorithm V takes as input

* the input x for L,
* ~~the empty~~ certificate y.

$V(x,y)$ does the following: run $A(x)$.

(thus: V ignores y)

$x \in L \iff A(x)$ returns YES

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

$\iff \exists$ certificate y such that

length of y = 0 = polynomial in the length of x,

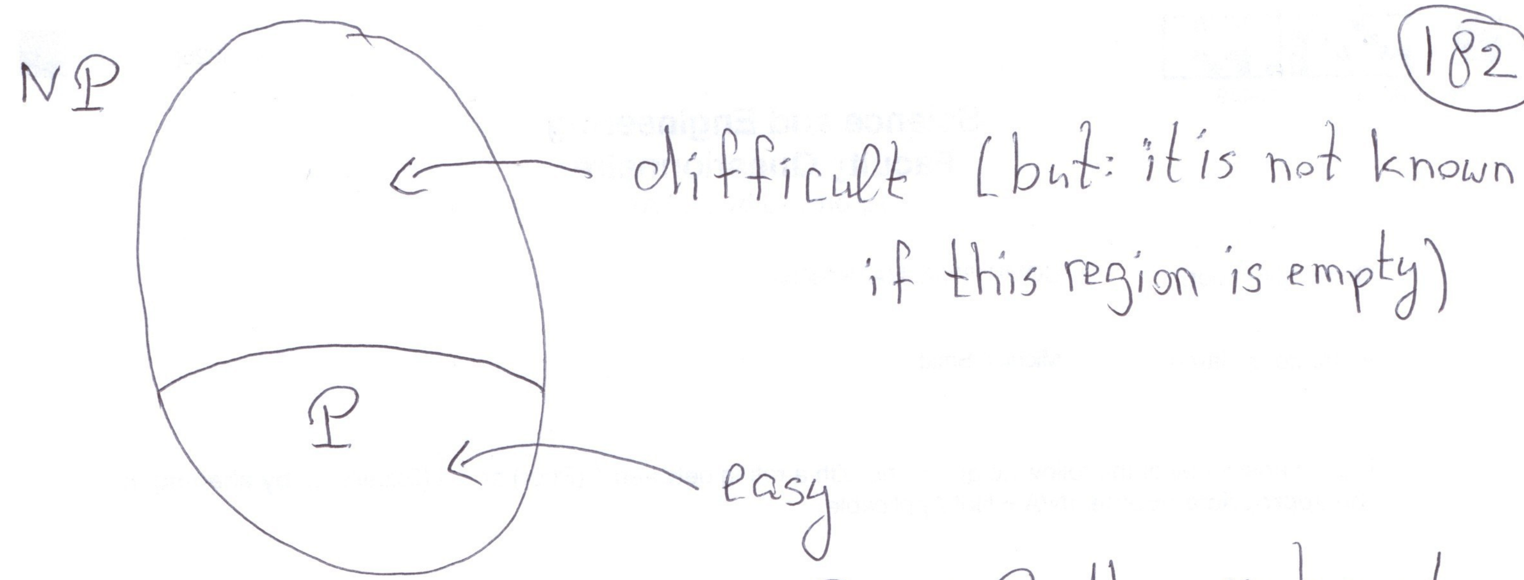
$V(x,y)$ returns YES

and running time of $V(x,y)$ = running time of $A(x)$ = polynomial in the length of x.

Therefore, L is in NP. $\square$

<u>Big Question</u>: 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

* $L \in NP$
* $L \notin P$

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.

$L \leq_P L'$ [L 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:

   $x \in L \iff f(x) \in$ [redacted] L'

3. for every input x to L:

   $f(x)$ can be computed in time that is polynomial in the length of x.