Let A be an algorithm that takes 2 bit strings x, y as input and returns

$$A(x,y) = \begin{cases} 0 & \text{"}y \text{ does not certify } x\text{"} \\ 1 & \text{"}y \text{ certifies } x\text{"} \end{cases}$$

Defn

we say y is a <u>certificate</u> for x iff $A(x,y)=1$. In this case we also say A <u>verifies</u> x. The <u>language verified by A</u> is

$$L = \{x \in \{0,1\}^* \mid \exists\, y \text{ such that } A(x,y)=1\}$$

i.e. A <u>verifies</u> L iff for all $x \in L$ there exists a certificate y for x.

consider again the concrete language HC consisting of (encoded) YES instances of Hamiltonian-Cycle. Its not difficult to write an algorithm that verifies this language. Given a bit string $\langle C \rangle$, check that

(1) C is a cycle, and
(2) C includes all vertices of G.

Thus with some effort

$$A(\langle G\rangle, \langle C\rangle) = 1$$

If C is a Hamiltonian cycle in G. (Here $\langle G\rangle$ and $\langle C\rangle$ denote the encodings of the graph G and cycle C as bit strings.) Furthermore this can be performed in Polynomial time.

Defn

$$NP = \{ L \subseteq \{0,1\}^* \mid \text{there exists a Polynomial time algorithm that verifies } L. \}$$

More Precisely, $L \in NP$ there exists a Polynomial time algorithm A and a constant c such that

$$L = \{ x \mid \exists y \text{ with } |y| = O(|x|^c) \text{ and } A(x,y) = 1 \}.$$

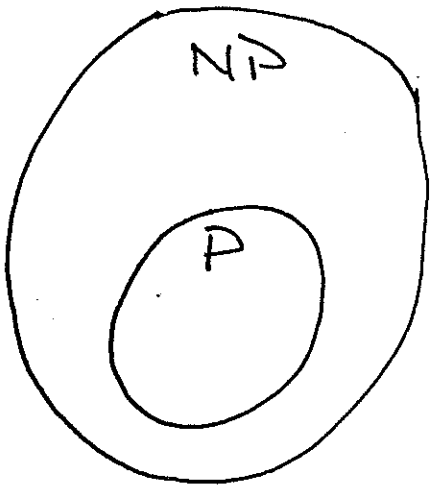
Thus, to say A is truely Polynomial time required that the size of the certificate y is Polynomially bounded by the size of x.

Observe that $P \subseteq NP$, since if we can decide a language in Polynomial time, we can certainly verify it in Polynomial time by just ignoring the certificate.

Ex.
$PATH \in P$ and hence $PATH \in NP$

Ex.
As we saw $HC \in NP$, but it is not known whether $HC \in P$ or $HC \notin P$

Picture:



Question: is $P = NP$ or is $P \neq NP$?

Defn

we say that L_1 is Polynomial time reducible to L_2 iff there exists a Polynomial time computable function

$$f : \{0,1\}^* \longrightarrow \{0,1\}^*$$

such that $x \in L_1$ iff $f(x) \in L_2$. In this case we write

$$L_1 \leq_P L_2$$

Lemma

If $L_1 \leq_P L_2$ then $L_2 \in P$ implies $L_1 \in P$. Likewise (by contraposition), If $L_1 \leq_P L_2$ then $L_1 \notin P$ implies $L_2 \notin P$.

Defn
we say L is NP-hard iff for every $L' \in NP$

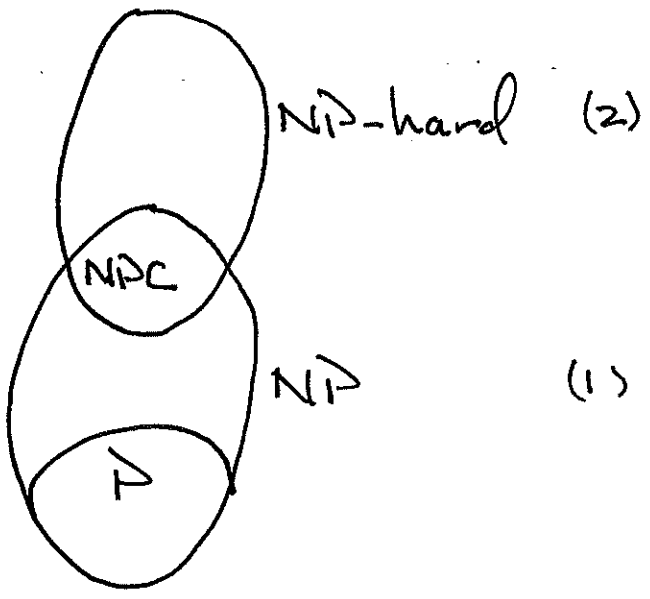
$$L' \leq_p L,$$

i.e. every NP language can be reduced (in polynomial time) to L.

Defn
we say L is NP-complete (NPC) iff

(1) $L \in NP$,
and (2) L is NP-hard

Picture if $P \neq NP$



Theorem
If there exists $L \in NPC \cap P$, then $P = NP$. If there exists $L \in NP - P$, then $NPC \cap P = \emptyset$.

Proof
Suppose $L \in NPC \cap P$. Pick any $L' \in NP$. By (2) of Defn. of NPC:

$$L' \leq_P L.$$

But since $L \in P$, we have $L' \in P$, by previous lemma. Since L' was arbitrary, we have

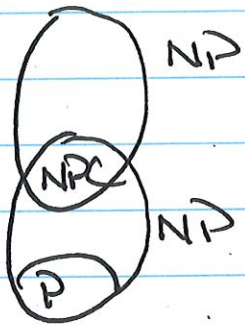
$$NP \subseteq P$$

whence

$$NP = P.$$

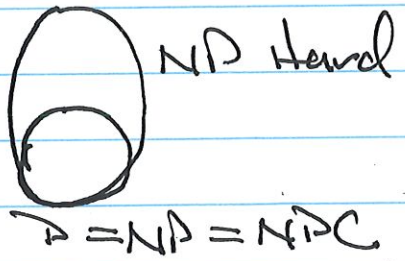
second statement follows. ∎

Thus we have two possible Pictures:

① 

$P \neq NP$

or

② 

$P = NP$

most researchers believe the true picture is ①.

2012 Poll:
- $P \neq NP$: 83 %
- $P = NP$: 9 %
- some form of undecidable : 4 %
- dont know or dont care : 4 %