# P, NP, PH, PSPACE

Note Title 9/28/2015

P is the class of languages accepted by polynomial-time Turing machines.

$$P = \bigcup_k TIME(n^k)$$

EXAMPLES

NP is the class of languages accepted by nondeterministic Turing machines in polynomial time

$$NP = \bigcup_k NTIME(n^k)$$

$P = NP$? we don't know

$HAM = \{ G = (V, E) : G \text{ has a Hamiltonian cycle} \}$

$SAT = \{ F : \exists x : F(x) = 1 \}$

$F(x) = (x_1 \wedge x_2 \vee \bar{x}_3) \wedge (\bar{x}_2 \vee x_3 \wedge x_4 \vee x_6) \ldots.$

Alternatively, NP is the class of languages for which membership ($x \in L$?) can be verified by a polynomial-time TM.
($\exists$ poly-size certificate).

EXAMPLES.

$P \subseteq NP \subseteq PSPACE \subseteq EXP$

$P \subsetneq EXP$ ↑ exponential time

from the Time hierarchy theorem.

That's all we know!

Let us explore further

$L \in NP \iff \exists NTM_M$ s.t. $L = \{x \mid M$ on input $x$ accepts$\}$ ($\exists$ path s.t.)

What about complements of languages in NP

$\overline{L} = \{x \mid x \notin L\}$

$= \{x \mid \nexists$ accepting path for $x\}$

$= \{x \mid \forall$ path . TM rejects$\}$ (accepts)

Co-NP is the class of languages that are complements of languages in NP.

Not clear how to check/verify $x \in L$ for $L \in$ co-NP

e.g. "G is not HAM"
"F is not satisfiable" } HOW TO CERTIFY?!

$SAT = \{F : \exists x \; F(x) = 1\} \quad \in NP$

$\overline{SAT} = \{F : \forall x \; F(x) = 0\} \quad \in co\text{-}NP$

$\Sigma_2 SAT = \{F : \exists x \; \forall y \; F(x, y) = 1\}$

$\Pi_2 SAT = \{F : \forall x \; \exists y \; F(x, y) = 0\}$

$\vdots$

$\Sigma_i SAT = \{F : \underbrace{\exists x_1 \; \forall x_2 \; \exists x_3 \ldots}_{i \text{ quantifiers}} \; F(x_1, \ldots x_i) = 1\}$

$\Pi_i SAT =$

Alternating Turing Machines have two types of nodes: (1) $\exists$ nodes which accept if some path from the node accepts and

(2) $\forall$ nodes _____ all paths _____ accept.

$\Sigma_i$ TM can alternate computation nodes $i$ times starting with $\exists$.

$\Pi_i$ _____ $\forall$.

$$PH = \bigcup_i \Sigma_i \bigcup_k TIME(n^k) = \bigcup_i \Pi_i \bigcup_k TIME(n^k)$$

$PH \subseteq PSPACE$

$$P \subseteq \underset{co\text{-}NP}{NP} \subseteq \underset{\Pi_2}{\Sigma_2} \subseteq \underset{\Pi_3}{\Sigma_3} \ldots \subseteq PSPACE.$$

How hard are problems in these complexity classes? Are some problems harder than others?

Let's go back to NP. Short proof of YES $x \in L$.

$k$-CLIQUE $= \{ G = (V, E), k \mid G$ has a clique of size $k \}$

TSP $= \{ G = (V, E)$, with distances $d_{ij}$ : $\exists$ tour of length $\leq D \}$

Integer LP $= \{ (A, b) : \exists x : Ax \leq b$ and $x \in \mathbb{Z}^n \}$

SAT $= \{ F : \exists x : F(x) = 1 \}$

Vertex-Cover $= \{ (G, k) : \exists S \subseteq V, |S| \leq k$ s.t. every edge has at least one endpoint in $S \}$

Ind-Set $= \{ ((G = V, E), k) : G$ has an ind. set of size $k \}$

Which of these problems is hard? harder?

Reduction from Problem A to Problem B
is an algorithm to solve an instance of Problem A using a blackbox procedure to solve Problem B.

KARP Reduction $X \longrightarrow Y$

$$X \in L_A \iff Y \in L_B$$

COOK Reduction can use many calls "$Y \in L_B$?" to decide if $X \in L_A$.

Polynomial-time reduction : # calls to oracle for B + additional computation time is polynomial.

Clique $\longrightarrow$ Ind. set

Ind. set $\longrightarrow$ clique'

Ind. set $\longrightarrow$ V.C.

Clique $\longrightarrow$ V.C.

SAT $\longrightarrow$ I.L.P.

Is there a <u>hardest</u> problem?

A language $L$ is called NP-complete if

1. $L \in NP$
2. $\forall A \in NP$, $\exists$ polytime reduction $A \longrightarrow L$.

Q. $\exists$ NP-complete problem?

A.

<u>Thm.</u> SAT is NP-complete.

[COOK/LEVIN]