

Computational Complexity.

→ How hard is it to solve problems?

problems: decision problems: input $x \in \{0,1\}^*$ is $x \in L$?

functions, search, ...

solve: Turing Machine.

Church-Turing thesis: TMs are "universal"

collection of states, tape (initially contains input x), transition function

$(state, tape) \rightarrow (state, write, move)$

\exists universal TM U given input i, x can simulate TM M_i on input x .

how hard: polynomial time is efficient n^{1000} ?

exponential time is not efficient. 2^n better? (for small n).

$n^{10} \rightsquigarrow n^3, n^4$.

"exact complexity" edit distance needs $\Omega(n^2)$ time. (assume ETH)

$O(n^2) = fn$ growing slower than n^2 faster
 $\Omega(n^2) =$ SAT needs $2^{n-O(n)}$

$P = \{L \mid \exists TM \text{ solving } L \text{ in polynomial time}\}$.

$NP = \{L \mid \exists NTM \text{ } \dots \}$ (def. 1)

$NP = \{L \mid \exists TM \ x \in L \text{ iff } \exists y \text{ TM verify } (x,y) \text{ in poly time}\}$ (def. 2).

why decision, not search?

SAT NP-complete problem.

formula φ over variables x_1, \dots, x_n .
 $x_i = 0$, simplify $\varphi' = \varphi(0, x_2, \dots, x_n)$ ✗
 $x_i = 1$, $\varphi'' = \varphi(1, \dots)$ ✗

$P=NP$ but no algorithm. only know runtime is $O(n^c)$.

for $t \in \mathbb{N}$:

for $i \in [1, t]$:

simulate TM_i for t many steps.

if accepts, produces witness y .

run verifier on (x, y) .

for SAT.

poly-time algorithm for SAT (assuming $P=NP$).

Impagliazzo "complexity worlds"?

$P=NP$

Intermediate problems between P & NP .

(before: padding).

thm: $EXP \neq NEXP \Rightarrow P \neq NP$.

↳ languages decidable in time 2^{n^c} for some c .

proof: assume $P=NP$. assume $L \in NEXP$ (M decides L).

$LPAD = \{ \langle x, 1 \dots 1 \rangle : x \in L \}$. $LPAD \in NP$.

M' simulates M on x \xrightarrow{NTM} hyp: $P=NP \exists M''$ solves $LPAD$ in poly time. \xrightarrow{DTM}

M''' simulates M'' runs in exp time, deterministic, solves L .

\xrightarrow{DTM}

$\langle x, 1 \dots 1 \rangle \rightsquigarrow x \mid 1 \dots 1$

$L \in NEXP = \bigcup_{c \in \mathbb{N}} NTIME(2^{n^c}) \exists c$.

Problems between P and NP .

Factoring, Graph Isomorphism.

thm (Ladner) If $P \neq NP$ then \exists problem not in P , not NP-hard.

$PADSAT = \{ \langle x, 1 \dots 1 \rangle : n = |x|, x \in SAT \}$.

$G(n) = \min i$ st $\exists x : |x| \leq \log n$ TM_i decide if $x \in PADSAT$ in time $\leq i \cdot n^i$

$H(n) = \min (G(n), \log \log n)$.

claim: if $PADSAT \in P$ then $H(n) = O(1)$ constant.
 if $\notin P$ $H(n) \rightarrow \infty$ (no bounded subsequence).

(i) $\exists TM_D$ solves $PADSAT$ in time n^c for some c .
 assume $w \log D \geq c$. without loss of generality.

$G(n) \in D, H(n) \in D$.

(ii) assume \exists bounded subseq S of $G(n) \in [1, M]$.

\exists subseq of $G(n)$ st $G(n) = C \in [1, M]$.

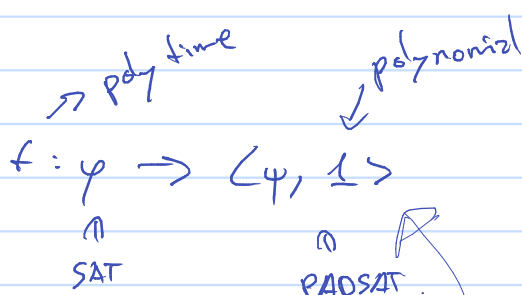
then TM_C can solve $PADSAT$ in time $C \cdot n^C$.

x , pick $n \geq 2^{|x|}$, pick element of subseq $> n$. // claim.

proof of thm: $P \neq NP$. $PADSAT \notin P$, $PADSAT$ not NP-hard.

(i) assume $PADSAT \in P$. by claim padding is n^c (i.e: polynomial).
 can solve SAT. in P !!

(ii) assume $PADSAT$ NP-hard



$PADSAT \notin P$ claim $n^{H(n)}$ not polynomial.