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) → (state, write, move)

$\exists$ universal TM U given input i, x can simulate TM $\#i$ 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 $\tilde{\Omega}(n^2)$ time. (assume ETH)

$O(n^2)$ = fn growing slower than n^2

$\Omega(n^2)$ = " " faster

SAT needs $2^{n-o(1)}$

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

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

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

why decision, not search?

SAT NP-complete problem.

formula φ over variables $x_1 \dots x_n$.

$x_1 = 0$, simplify $\varphi' = \varphi(0, x_2 \dots x_n)$ $\forall$

$x_1 = 1$, $\varphi'' = \varphi(1 \quad)$ $\forall$

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

→ for SAT.

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).

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 (NTM) decides L).

$LPAD = \{\langle x, 1 \dots 1 \rangle : x \in L\}$ (2^{n^c} ones). $LPAD \in NP$.

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

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

$\langle x, 1 \dots 1 \rangle \rightsquigarrow x\,0\,1 \dots 1$

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

Problems between P and NP.

Factoring, Graph Isomorphism.

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

$PADSAT = \{\langle x, 1 \dots 1 \rangle$ ($n^{H(n)}$ ones) $: n = |x|, x \in SAT\}$.

$G(n) =$ min i s.t. $\forall x$: $|x| \leq \log n$ 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 $\notin P$ $H(n) \to \infty$ (no bounded subsequence).

(i) $\exists TM_D$ solves PADSAT in time n^c for some C.

assume wlog $D \geq C$. without loss of generality.

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

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

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

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 $\notin P$, (i) PADSAT not NP-hard (ii)

(i) assume PADSAT $\in P$. by claim padding is n^C (i.e. polynomial).

can solve SAT. in P !!

(ii) assume PADSAT NP-hard

$f: \varphi \to \langle \varphi, 1 \rangle$ (poly time; polynomial)

φ: SAT; $\langle \varphi, 1 \rangle$: PADSAT.

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