UNIFORM COMPUTATION

Ch 9. **Restricted Computational Models** | **Regular Expressions** | **Def.** A regex e over an alphabet Σ is a str over $\Sigma \cup \{(,),|,*,\emptyset,\text{""}\}$ that has form: (1) $e=\sigma$, where $\sigma \in \Sigma$, (2) $e=(e'|e'')$, (3) $e=(e')(e'')$, (4) $e=(e')^*$, (5) $e=\emptyset$ no strings, (6) $e=$ "" empty string. (REG $\subsetneq$ TIME($O(n)$) $\subseteq$ P) | **Matching.** $\phi_e: \Sigma^* \to \{0,1\}$, (1) $\phi_e(x)=1$ iff $x=\sigma$. (2) $\phi_e(x)=\phi_{e'}(x) \vee \phi_{e''}(x)$. (3) $\phi_e(x)=1$ iff $\forall x', x''$ s.t. $x=x'+x''$ and $\phi_{e'}(x')=\phi_{e''}(x)=1$. (4) $x=x_0 \cdots x_{k-1}$ and $\phi_{e'}(x_i)=1$ for every $i \in [k]$. | Thm 9.5. $\forall e$ over $\{0,1\}$, $\phi_e: \{0,1\}^* \to \{0,1\}$ is computable. | **Proofs** Induction over the length m-ish of e, aka # of symbols. | **Most efficient matching algo is $O(n)$.** | **DFA** Def. DFA w/ C states over $\{0,1\}$ is a pair (T,S) w/ transition func $T: [C] \times \{0,1\} \to [C]$ and set of accepting states $S \subseteq [C]$. (T,S) computes $F: \{0,1\}^* \to \{0,1\}$ if $\forall n \in \mathbb{N}$, $x \in \{0,1\}^n$, $v_0=0$ and $v_{i+1}=T(v_i, x_i)$, $\forall i \in [n]$, $v_n \in S \leftrightarrow F(x)=1$. | **DFA-regex equivalence.** F is regular iff $\exists$ DFA (T,S) that computes F. | **Pumping lemma** If exp e computes F then there exists n_0 s.t. $\forall w$ with $F(w)=1$ and $|w| \geq n_0$, there $\exists$ partition $w=xyz$ with $|xy| \leq n_0$, $|y| \geq 1$ s.t. $\forall k \in \mathbb{N}$ it holds that $F(xy^k z)=1$. | **Usage** (1) Choose some arbitrary n_0 (whose existance is guaranteed by pumping lemma) (2) choose $w=0^{n_0}1^{n_0}$ (choose any k. $k=2$.) (3) Since $|xy| \leq n_0$ and $|y| \geq 1$, forced to use $x=0^a$, $y=0^b$, $z=0^{n_0-a-b}1^{n_0}$ for $b \geq 1$, $a \leq n_0-b$. (4) $xy^kz=0^{n_0+b}1^{n_0}$, $F(xy^2z)=0 \to$ contradiction.

computable?	halting	emptiness	equivalence
regexp	✓	✓	✓
context-free	✓	✓	X
turing-compl	×	×	×

EFFICIENT COMPUTATION

Ch 12. **Modeling Running Time** | **Running Time** $T: \mathbb{N} \to \mathbb{N}$. $F: \{0,1\}^* \to \{0,1\}^*$ is computable in $T(n)$ Single-Tape-TM time if $\exists$ TM M s.t. for every sufficiently large n and every $x \in \{0,1\}^n$, M halts on input x after at most $T(n)$ steps and outputs $F(x)$. $\text{TIME}_{TM}(T(n))$ = set of bool funcs computable in $T(n)$ TM time. TIME is class of functions. | **NAND-RAM** same except at most $T(n)$ lines. | **RAM $\leftrightarrow$ TM** $n \mapsto T(n)$ can be computed by TM in $O(T(n))$. $\text{TIME}_{TM}(T(n)) \subseteq \text{TIME}_{RAM}(10 \cdot T(n)) \subseteq \text{TIME}_{TM}(T(n)^4)$ | **P** $F: \{0,1\}^* \to \{0,1\} \in P$ if $\exists$ polynomial $p: \mathbb{N} \to \mathbb{R}$ and TM M s.t. $\forall x \in \{0,1\}^*$, M halts at at most $p(|x|)$ steps and outputs $F(x)$. | **Ex:** Shortest path, MINCUT, 2SAT, Linear Eqs, 2-colorable, Determinant, Primality. | **EXP** Same as P except halts within at most $2^{p(|x|)}$ steps. | **Ex:** Longest path, MAXCUT, 3SAT, Quad exps, Nash, Permanent, Factoring. | **Efficient Universal Machine** $\exists$ NAND-RAM prog U s.t: (1) U is universal: NAND-RAM program $U(P,x)=P(x)$. (2) U is efficient: $\exists a,b$ s.t. $\forall P$, if P halts on x after at most T steps, $U(P,x)$ halts after at most $c \cdot T^a$ steps, where $c \leq a|P|^b$. | **Timed Universal TM** Let $\text{TIMEDEVAL}: \{0,1\}^* \to \{0,1\}^*$ be defined as $\text{TIMEDEVAL}(M,x,1^T) = \begin{cases} M(x) & M \text{ halts within} \leq T \text{ steps on } x \\ 0 & \text{o/w} \end{cases}$, then TIMEDEVAL $\in P$. (Thm 12.8) | **Time Hierarchy Thm.** Def For every nice function $T: \mathbb{N} \to \mathbb{N}$, there $\exists F: \{0,1\}^* \to \{0,1\}$ in $\text{TIME}(T(n)\log(n)) \setminus \text{TIME}(T(n))$ (any efficiently computable func that tends to ∞) | **Corollary.** $P \subsetneq EXP$. Ex: $F(M,x)=1$ if M halts w/ at most $|x|^{\log|x|}$ steps. (Bounded HALT) | Note: $P \subsetneq \text{TIME}(n^{\log n}) \subsetneq \text{TIME}(2^n) \subsetneq \text{TIME}(2^{n^2}) \subsetneq EXP \subsetneq \text{TIME}(2^{2^n})$.

Non-uniform computation. Def $F: \{0,1\}^* \to \{0,1\}$, $T: \mathbb{N} \to \mathbb{N}$ is a nice bound. $\forall n \in \mathbb{N}$, $F_n: \{0,1\}^n \to \{0,1\}$ is restriction of F to n. F is non-uniformly computable in at most $T(n)$ size, $F \in \text{SIZE}(T(n))$ if $\exists$ sequence $(C_0, C_1, \ldots)$ of NAND circuits s.t. (1) $\forall n \in \mathbb{N}$, C_n computes F_n. (2) for sufficiently large n, C_n has at most $T(n)$ gates.

P/poly $= \bigcup_{c \in \mathbb{N}} \text{SIZE}(n^c)$. **$P \subsetneq P/poly$** (Thm 12.12 → unroll NAND-TM to NAND-CIRC) There is some $a \in \mathbb{N}$ s.t. $\forall$ nice $T: \mathbb{N} \to \mathbb{N}$ and $F: \{0,1\}^* \to \{0,1\}$, $\text{TIME}(T(n)) \subseteq \text{SIZE}(T(n)^a)$.

P/poly has uncomputable functions UHALT: $S: \mathbb{N} \to \{0,1\}^*$ outputs binary rep of input w/o most significant 1 (onto function). $\forall x \in \{0,1\}^*$, $\text{UHALT}(x) = \text{HALTONZERO}(S(1x1))$. UHALT always $=1$ or always $=0$ → NAND-TM rep exists w/ const # of lines. **Unknown** if EXP $\subseteq$ P/poly? Believed that EXP $\not\subseteq$ P/poly.

Ch 13. **Polynomial-time Reductions** | **Reduction Format.** | **Proof that $F \in NP$** (1) What a valid certificate w for an input x looks like. (2) What the verifying program V does (how $V(w,x)=1$). (3) Why V is poly-time. (4) Why x has a valid witness iff $F(x)=1$ → $\exists w: V(w,x)=1$ implies $F(x)=1$. | **Proof that F is NP hard** Reduce from NP-hard H. Reduce H to F: $H \leq_p F$. (1) High-level implementation of transformation (program) R, which transforms inputs of H to inputs of F $x'=R(x)$. (2) Perform runtime analysis of R to show that it is polynomial time wrt $|x|$, len of input. (3) Show correctness to prove $H(x)=F(R(x))$, or $H(x)=F(x')$. (a) Completeness: $H(x)=1 \to F(R(x))=1$. Assume some arbitrary input x s.t. $H(x)=1$ apply transformation $x'=R(x)$ and show by definition of F, that $F(x')=1$. (b) Soundness: show that $F(R(x))=1 \to H(x)=1$ (or contrapositive, $H(x)=0 \to F(R(x))=0$). Assume some x' s.t. $F(x')=1$, reverse transformation, show by def of H, any x that would have been transformed into x' must be s.t. $H(x)=1$. (→ for every x' assignment, it came from an x)

NP-Hard Problems 3SAT: $\{0,1\}^* \to \{0,1\}$ w/ 3CNF formula φ as input, mapped to 1 if $\exists$ some assignment to vars of φ that make it true. m clauses, vars can be negated. MAXCUT: $\{0,1\}^* \to \{0,1\}$ maps graph G and number k to 1 if $\exists$ cut S ($\forall$ edges cut, only one end vertex in S) that cuts at least k edges. SUBSETSUM: Input list of #s $x_0 \cdots x_{n-1}$, $t \in \mathbb{N}$. Output 1 iff $\exists$ set $S \subseteq [n]$ s.t. $\sum_{i \in S} x_i = t$. ($O(n)$ if $|x_i|=k$ bits → algo that runs in poly $(n, 2^k)$) | **Integer Programming** Input list of m linear inequalities in n variables $x_0 \cdots x_{n-1}$ with each inequality of the form $a_0x_0+a_1x_1+\cdots+a_{n-1}x_{n-1} \geq b$. Output 1 if $\exists$ a solution $x_0 \cdots x_{n-1} \in \mathbb{Z}^n$ to the inequalities. (input len $= \sum |a_i| + |b|$ for all ints) (general form of 01 LINEQ) | **ISET** Input: graph G and independent set size k. Output 1 if $\exists$ subset S s.t. no edges w/ both endpoints in S of at least k. | **VERTEXCOVER** Input: $G=(V,E)$ and number k. Output 1 if $\exists$ set S of $|S| \leq k$ s.t. $\forall$ edge (u,v), either $u \in S$ or $v \in S$. | **3COLOR** Input: $G=(V,E)$. Output 1 if there $\exists$ assignment of color to each vertex s.t. vs connected by edges are not same color.

Reduction Strategies. 3SAT Auxiliary variables. Ex: 4SAT → 3SAT, $(a_1 \vee a_2 \vee z) \wedge (a_3 \vee a_4 \vee \bar{z})$. If they're supposed to be the same, add $(x_i \vee \bar{x}_i') \wedge (\bar{x}_i \vee x_i')$ clauses to force assignment. **Unconnected vertices** to a graph (n) **Invert** the graph **List of Pairs** → edges of a graph, let x_is be the vertices. **Edges of graph** → list of pairs, vertices = people **clauses** ↔ Eqns $\neg x_i = y_i$, add $x_i + y_i = 1$ (only one can be one).

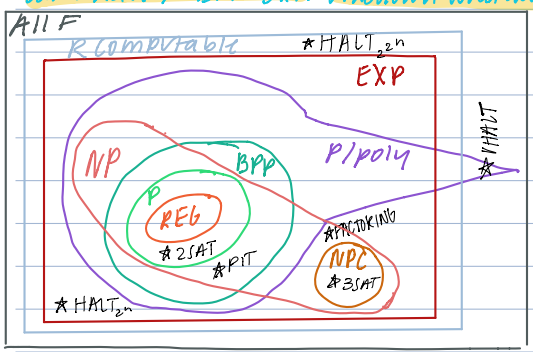
Quantifier Elimination

Ch 14. NP, NPC, and Cook-Levin **NP** $F:\{0,1\}^* \to \{0,1\}$ is in NP if $\exists$ int $a>0$ and $V:\{0,1\}^* \to \{0,1\}$ s.t. $V \in P$ and for every $x \in \{0,1\}^n$, $F(x)=1 \leftrightarrow \exists_{w \in \{0,1\}^{n^a}}$ s.t. $V(xw)=1$. **NP not (necessarily) closed under complement** $\exists F$ s.t. $F \in NP$ and $1-F \notin NP$. For P, if $F \in P$, $1-F \in P$. **$P \subseteq NP$** verifier algo = solving algo. **$NP \subseteq EXP$** enumerate over all 2^{n^a} strings and eval in poly(n) time. **$P \subseteq NP$ or $NP \subseteq EXP$ is strict** by Time Hierarchy. $\overline{3SAT(\varphi)} = 1 - \overline{3SAT(\varphi)}$ believed to be $\in EXP / NP$. If $F \leq_p G$ and $G \in NP$, then $F \in NP$. **NP-Hard** $G:\{0,1\}^* \to \{0,1\}$ is NP-hard if for every $F \in NP$, $F \leq_p G$. **NP-Complete** G is NP-hard and $G \in NP$. **Ladner's Thm** if $P \neq NP$, then $\exists$ problems in NP that are not in P and are not NPC. **Cook-Levin Theorem** For every $F \in NP$, $F \leq_p 3SAT$. **Outline** $F \leq_p NANDSAT \leq_p 3NAND \leq_p 3SAT$. **Implications** Finding a poly-time algo for any NPC problem implies a poly-time algo for all. Either All NPC $\subseteq P$ or no NPC $\subseteq P$. **If one NPC problem in $P_{/poly}$, then NPC $\subseteq P_{/poly}$.** Ch 15. **What if P=NP?** **Search vs. Decision** Theorem 15.1 ← can find satisfying assignment in poly-time. Suppose P=NP. $\forall$ poly-time algo V and $a, b \in \mathbb{N}$, there is a poly-time FINDv s.t. $\forall x \in \{0,1\}^n$, $\exists y \in \{0,1\}^{an^b}$ satisfying $V(x,y)=1$, then FINDv(x) finds some string y' satisfying this condition. **Proof.** STARTSWITHv, on input $x \in \{0,1\}^*$ and $z \in \{0,1\}^\ell$, outputs 1 iff $\exists y \in \{0,1\}^{an^b}$ s.t. first ℓ bits of $y = z$. FINDv calls STARTSWITHv $2 \cdot an^b$ times. **Optimization** ← s2d finds optimal solution Suppose that P=NP. $\exists$ poly-time computable function $f:\{0,1\}^* \to \{0,1\}^*$ there is a poly-time algo OPT s.t. on input $x \in \{0,1\}^*$, $OPT(x, 1^m) = \max_{y \in \{0,1\}^m} f(x,y)$. Similarly, $\exists$ poly-time algo FINDOPT s.t. $\forall x \in \{0,1\}^*$, $FINDOPT(x, 1^m)$ outputs $y^* \in \{0,1\}^m$ s.t. $f(x,y^*) = OPT(x, 1^m)$. **Proof.** (can test in poly($|x|, m$) time if $\exists y$ s.t. $f(x,y) \geq k$. Binary search. **Cryptography** If P=NP, then almost every cryptosystem can be efficiently broken. **Finding Mathematical Proofs** $V(x,w)=1$ iff w encodes a valid proof for statement x. Implies SHORTPROOF$_V(x, 1^m)=1$ if $\exists w \in \{0,1\}^*$ with $|w| \leq m$ s.t. $V(xw)=1$. If P=NP, despite Gödel's Incompleteness Thm, automate finding proofs.

RANDOMIZED COMPUTATION

Ch 17. Prob Theory 101 **RVs** $E[X] = \sum_{x \in \{0,1\}^n} 2^{-n} X(x)$ **Linearity of Expectation** $E[X+Y] = E[X] + E[Y]$ **Union Bound** for every A, B: $Pr[A \cup B] \leq Pr[A] + Pr[B]$ **Independence** Def Ind. if $Pr[A \wedge B] = Pr[A] \cdot Pr[B]$. **Disjoint** Separate from ind. if $A \wedge B = \emptyset$. **Positively correlated** $P[A \wedge B] \geq Pr[A] \cdot Pr[B]$. **Conditional Prob.** $Pr[B|A] = Pr[A \wedge B] / Pr[A]$. **More disjoint** If X, Y are rvs that depend on two disjoint sets of coins S, T, then they are independent. $E[\prod_{i=0}^{n-1} X_i] = \prod_{i=0}^{n-1} E[X_i]$. Functions also preserve independence. **Concentration / Tail Bounds** **Markov's Inequality** If X is a non-negative rv then $Pr[X \geq k E[X]] \leq 1/k$. Not tight **Chebyshev's Inequality** Suppose that $\mu = E[X]$ and $\sigma^2 = Var[X]$. Then for every $k>0$, $Pr[|X-\mu| \geq k\sigma] \leq 1/k^2$. Used w/ $X = X_1 + X_2 + \cdots X_n$ iid rv's $\in [0,1]$. **The Chernoff Bound** w/ large n, distribution becomes ~ Normal, probability of deviating exponentially decays. If $X_1, \ldots X_n$ are iid rvs s.t. $X_i \in [0,1]$ and $E[X_i] = p$ for every i, then for every $\epsilon > 0$, $Pr[|\sum_{i=0}^{n-1} X_i - pn| > \epsilon n] \leq 2 \cdot e^{-2\epsilon^2 n}$. Ch 18. **Probabilistic Computation.** **Maxcut Approx.** $\exists$ efficient probabilistic algo that on input of n-vertex, m-edge G, outputs cut (S,T) that cuts at least m/2 edges in expectation. **Amplification Idea** Repeat process several times and output best cut. Use inequality ★19.5 $(1-1/k)^k \leq 1/e \leq 1/2$. W/ 2000m samples, prob of failure is at most $(1-1/(2m))^{2000m} \leq 2^{-1000}$ **One-sided Error** Only one can happen: Prog P may output 1 when F(x)=0, or P may output 0 when F(x)=1. **Two-sided Amplification** **Two-sided error** both can happen. Cannot amplify by repeating k times and output 1 if ≥ 1. **Thm** If $F:\{0,1\}^* \to \{0,1\}$ is a function s.t. $\exists$ poly-time algo A satisfying $Pr[A(x) = F(x)] \geq 0.51$, $\forall x \in \{0,1\}^*$ then there $\exists$ poly-time algo B satisfying $Pr[B(x) = F(x)] \geq 1 - 2^{-|x|}$ for every $x \in \{0,1\}^*$.

Ch 19. **Modelling Randomized Computation** **BPP** **Def.** $F \in BPP$ if $\exists a, b \in \mathbb{N}$ and RNAND-TM program P s.t. $\forall x \in \{0,1\}^*$, on input x, P halts within at most $a|x|^b$ steps and $Pr[P(x) = F(x)] \geq 2/3$ where this prob is taken over the result of the RAND operations of P. ← can amplify arbitrarily close to 1. not over choice of input x. Must be $\geq 2/3$ on every possible input. **Alt Def.** $F \in BPP$ if $\exists a, b \in \mathbb{N}$ and $G:\{0,1\}^* \to \{0,1\}$ s.t $G \in P$ and $\forall x \in \{0,1\}^*$, $Pr_{r \sim \{0,1\}^{a|x|^b}}[G(xr) = F(x)] \geq 2/3$. **NP vs. BPP** NP is one-sided $F(x)=1$ if $\exists$ solution w s.t. $G(xw)=1$ and $F(x)=0$ if for every str w, $G(xw)=0$. BPP is symmetric wrt $F(x)=0$, $F(x)=1$. **NP is intuitive** cannot actually calc SOL in exp many possibilities. BPP can compute F in practice. **Amplification** F is bool funct s.t. there is poly $p: \mathbb{N} \to \mathbb{N}$ and poly-time randomized algorithm A s.t. $\forall x \in \{0,1\}^n$, $Pr[A(x) = F(x)] \geq \frac{1}{2} + \frac{1}{p(n)}$. Then $\forall$ poly $q: \mathbb{N} \to \mathbb{N}$ $\exists$ poly-time rand alg B s.t. $\forall x$, $Pr[B(x) = F(x)] \geq 1 - 2^{-q(n)}$. **NP-hard and BPP** ≡ if 3SAT $\in$ BPP if F is NP-hard and $\in BPP$, then $NP \subseteq BPP$. We get essentially all P=NP consequences. **Unknown** if BPP=EXP. **Unknown** how to derandomize **Unknown** if BPP=P. **Suspected** if true, supports extended Church-Turing hypothesis that deterministic poly-time algos capture feasibly computable. **Suspected** $BPP \subseteq EXP$. Brute through all r inputs ($2^{a|x|^b} \cdot$ poly) and maj vote. **$P \subseteq BPP$.** Throw away randomness. **$BPP \subseteq P_{/poly}$** ← if 3SAT $\notin P_{/poly}$ then NP $\not\subseteq$ BPP (widely believed) $\forall x$, can show $\exists r$ s.t. it is correct for all inputs (but don't know what it is) via amplification. but conj: existence of pseudorandom generators. **Can't have P=BPP=EXP.** **Unknown whether BPP$\subseteq$NP, NP$\subseteq$BPP, or incomparable.** **Sipser-Gács Theorem.** If P=NP, then BPP=P.



Ladner's Thm: $P \neq NP \to \exists F \in NP \setminus P$ and F is not NPC. **SIZE(T):** set of all functions $F:\{0,1\}^* \to \{0,1\}$ s.t. for every (large enough n, the restriction of F to $\{0,1\}^n$ has circuit of at most T gates. Has uncomputable funcs.

	P	BPP	NP	EXP	P/poly	
P		P⊆BPP	P⊆NP	P⊆EXP	P⊆P/poly	confirmed T
BPP	BPP⊆P		BPP⊆NP	BPP⊆EXP	BPP⊆P/poly	confirmed F
NP	NP⊆P	NP⊆BPP		NP⊆EXP	NP⊆P/poly	unknown
EXP	EXP⊆P	EXP⊆BPP	EXP⊆NP		EXP⊆P/poly	BPP=P P=NP
P/poly	P/poly⊆P	P/poly⊆BPP	P/poly⊆NP	P/poly⊆EXP		BPP=EXP