Chapter 1. Basic Function Properties. (1) $F: S \to T$ and $G: T \to U$ are one-to-one, then $H: S \to U$ $H(s) = G(F(s))$ is one-to-one. (2) $F: S \to T$ is one-to-one, onto $G: T \to S$ exists where $G(F(s)) = s$ for every $s \in S$. (3) If $G: T \to S$ is onto then $F: S \to T$ exists where $G(F(s)) = s$. (4) Finite S, T: (a) $|S| \leq |T|$, (b) one-to-one $F: S \to T$, (c) onto $G: T \to S$. Big-O Notation. For $F, G: \mathbb{N} \to \mathbb{R}_+$, $F = O(G)$ if $a, N_0 \in \mathbb{N}$ s.t. $F(n) \leq a \cdot G(n)$. $F = \Theta(G)$ if $F = O(G)$ and $G = O(F)$. $F = \Omega(G)$ if $G = O(F)$. Little-O. $F = o(G)$ if for every $\epsilon > 0$ there is some N_0 s.t. $F(n) < \epsilon G(n)$ for every $n > N_0$. $F = \omega(G)$ if $G = o(F)$. Topological Sorting. G is directed graph. G is acyclic iff there exists a layering of G.

Chapter 2. Computation and Representation. Cantor's Theorem. The $\mathbb{R}$ are uncountable. There does not exist a one-to-one mapping $\mathbb{R} \to \{0,1\}^*$. Cantor Power Set. If A is any set, then $|A| < |P(A)|$. $\{0,1\}^* \to \mathbb{N}$ one-to-one. $\{0,1\}^\infty$ is $\{f: \mathbb{N} \to \{0,1\}\}$. $\{0,1\}^\infty \to \mathbb{R}$ one-to-one. Prefix-Free Encoding. For two strings y, y', we say that y is a prefix of y' if $|y| \leq |y'|$ and for every $i < |y|$, $y_i = y'_i$. Function E is prefix-free if $0, 0' \in 0$ do not exist s.t. $E(0)$ is prefix of $E(0')$. (↳ must be one-to-one) len(encoding) $= 2\lceil \log n \rceil + 2$

FINITE COMPUTATION

Chap 3. Defining Computation. Algorithm. Set of instructions for how to compute an output from input by following steps of "elementary steps". AON Properties. (1) commutivity, (2) associativity $(a \wedge b) \wedge c = a \wedge (b \wedge c)$, (3) distributive $a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$. IF. $= \text{OR}(\text{AND}(a,b), \text{AND}(\text{NOT}(a), c))$. XOR. $(a,b) = \text{NAND}(\text{NAND}(a, \text{NAND}(a,b)), \text{NAND}(b, \text{NAND}(a,b))) = \text{OR}(\text{AND}(a, \text{NOT}(b)), \text{AND}(\text{NOT}(a), b))$. $1 = \text{OR}(a, \text{NOT}(a))$ $0 = \text{AND}(a, \text{NOT}(a))$

Universal: AND/NOT, OR/NOT, NOR = NOT(X OR Y), XOR/AND. Circuits. $f: \{0,1\}^n \to \{0,1\}^m$. C computes f if for every $x \in \{0,1\}^n$, $C(x) = F(x)$. Circuits ↔ Straight Line Programs. $f: \{0,1\}^n \to \{0,1\}^m$, $s \geq m$, f is computable by Boolean circ of s gates iff f computable by AON-CIRC of s lines. NAND. $\text{NOT}(a) = \text{NAND}(a,1)$. $\text{AND}(a,b) = \text{NAND}(\text{NAND}(a,b), \text{NAND}(a,b))$. $\text{OR}(a,b) = \text{NAND}(\text{NAND}(a,1), \text{NAND}(b,1))$. For every bool circ C w/ s gates, NAND-CIRC C' has at most $3s$ gates. Equivalence. NAND ≡ {IF, ZERO, ONE}. AON-CIRC ≡ IF-CIRC. Neither {AND, OR} or {XOR} is more powerful. {OR, NOT} ≡ NAND. NAND > XOR. {MAJ, NOT, ONE} ≡ NAND. NOR ≡ NAND.

Chap 4. Syntactic Sugar. [functions/macros, conditionals, bounded loops] → LOOKUP → compute every function. NAND-CIRC-PROC. defines non-recursive procedures. $\text{IF}(\text{cond}, a, b) = \text{NAND}(\text{NAND}(b, \text{NAND}(\text{cond}, \text{cond})), \text{NAND}(a, \text{cond}))$. LOOKUP Function. For every k, $\text{LOOKUP}_k: \{0,1\}^{2^k + k} \to \{0,1\}$ is $\forall x \in \{0,1\}^{2^k}$, $i \in \{0,1\}^k$, $\text{LOOKUP}_k(x, i) = x_i$. Thm 4.10 For every $k > 0$, ∃ NAND-CIRC prog that computes LOOKUP (w/ at most $4 \cdot 2^k$ lines). Universality of NAND. Thm 4.12. ∃ c s.t. $\forall n, m > 0$, $f: \{0,1\}^n \to \{0,1\}^m$, NAND-CIRC prog w/ at most $c \cdot m \cdot 2^n$ lines computes f. Improved bound: $O(m \cdot 2^n/n)$. Proof w/ LOOKUP. ∴ Every finite function computable w/ NAND-CIRC. SIZE(T) Def. $\text{SIZE}_{n,m}(s)$ = set of finite functions s.t. $f \in \text{SIZE}(s) = \bigcup_{n,m \leq 2^s} \text{SIZE}_{n,m}(s)$ = all functions w/ NAND-CIRC at most s lines. ↳ restriction F to $\{0,1\}^n$ has circuit of at most T gates. Thm 4.16 $|\text{SIZE}(s)|$ is smaller than the total # of functions mapping n bits to 1 bit. Upper Bound $\forall n, m > 0$, $f: \{0,1\}^n \to \{0,1\}^m$, $\leq c \cdot m \cdot 2^n/n$ bool circ lines, $c \leq 100$, arbitrary small. Size Hierarchy Thm. For every large n and $10n < s < 0.1 \cdot 2^n/n$, $\text{SIZE}_n(s) \subsetneq \text{SIZE}_n(s+10n)$. $\text{SIZE}_{n,1} \subsetneq \text{SIZE}_{n,1}(n^3) \subsetneq \text{SIZE}_{n,1}(100 \cdot 2^n/n) = \text{ALL}_{n,1}$

Chap 5. Code as Data, Data as Code. Programs ↔ Strings for $f \in \text{SIZE}(s)$, ∃ P computing f whose string rep has $c \cdot s \log s$ lines. $c = 10$. Counting Programs $\forall s \in \mathbb{N}$, $|\text{SIZE}(s)| \leq 2^{c \cdot s \log s}$ ($c = 10$) functions computable by an s-line NAND-CIRC program. ($f \notin \text{SIZE}(\frac{\delta 2^n}{n})$ there exists a) Program ↔ # lines lower Bound For sufficiently large n, $f: \{0,1\}^n \to \{0,1\}$'s NAND-CIRC program requires at least $\delta \cdot 2^n/n$ lines ($0.1 < \delta \sim 1/2$). TUPLE Representation n inputs, m outputs: (n, m, L). L = list of tuples of NAND gates. Universality Thm 5.10: Universality of NAND-CIRC $\forall s, n, m \in \mathbb{N}$ ∃ NAND-CIRC prog of at most $O(s \log s)$ lines that computes $\text{EVAL}_{s,n,m}: \{0,1\}^{|s|_n} \to \{0,1\}^m$. $\text{EVAL}_{s,n,m}$

$\text{EVAL}_{s,n,m}(P, x) = \begin{cases} P(x) & P \in \{0,1\}^{S(s)} \\ 0^m & \text{otherwise} \end{cases}$, where $S(s) = 3s\lceil \log(3s) \rceil$ ← length of triple notation. Physical Extended Church-Turing Hypothesis If a function $F: \{0,1\}^n \to \{0,1\}^m$ can be physically computed w/ s resources ↔ boolean circuit w/ s gates.

UNIFORM COMPUTATION

Chap 6. Loops and Infinity. Turing Machines. Computable Functions. $F: \{0,1\}^* \to \{0,1\}^*$, TM M. M computes F if $\forall x \in \{0,1\}^*$, $M(x) = F(x)$. F is computable if there ∀ a TM that computes it. NAND-TM ≡ NAND CIRC + loops (arbitrary time) + Arrays (arbitrary sequences). NOOAND JUMP (last line): $a = 0, b = 0$: halt. $a = 1, b = 0$: jump to first. $a = 0, b = 1$: jump to ith line and i--. $a = 1, b = 1$: jump to first line and i++. (i is integer valued.)

Turing Equivalence. TMs ≡ NAND-TM programs: $\forall F: \{0,1\}^* \to \{0,1\}^*$, F is computable by a NAND-TM program iff there is a TM that computes it. Uniformity. A single algo can compute functions of all input lengths.

Chap 7. Equivalent Models of Computation. RAM Machines. Allows for directly accessing memory locations. Can do: data movement, computation, and control flow. NAND-RAM ≡ NAND TM + integer valued variables + indexed arr access + if/then + while/do. TM ↔ RAM Machines Thm 7.1 $\forall F$, F is computable by a NAND-TM iff F is computable by a NAND-RAM program. Pf: Indexed access of bit arrays → 2D bit arrays → arrays of ints

Turing Equivalent Models. TM, NAND-TM, NAND-RAM, λcalc, Game of Life, Programming languages (Python, C, etc.). NAND-TM Config. fixed len w/ var and blocks (active, final, initial, vars, line) Configuration of TMs. Let M be a TM w/ tape alphabet Σ and state space $[k]$. A configuration of M is a string $\alpha \in \tilde{\Sigma}^*$ where $\tilde{\Sigma} = \Sigma \times (\{\cdot\} \cup [k])$ that satisfies that there is exactly 1 coordinate i for which $\alpha_i = (\sigma, s)$ for some $\sigma \in \Sigma$ and $s \in [k]$. For all other coordinates j, $\alpha_j = (\sigma', \cdot)$. A configuration $\alpha \in \tilde{\Sigma}^*$ of M corresponds to the following state of its execution: (1) since α_j is a pair of alphabet symbol σ and either a state in $[k]$ or $\cdot$, $\alpha_{j,0}$ is the first component σ of this pair. (2) M's head is in the unique position i for which α_i has the form (σ, s) for $s \in [k]$ and M's state = s.

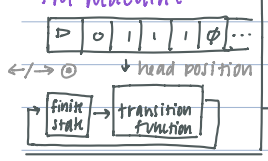
Axa: has (1) current head position, (2) full contents of large-scale memory (tape), (3) content of local registers (state of M). NAND-TM Config: (1) current value of i (2) for every scalar var foo, the value of foo (3) for every arr Bar, Bar[j] for every $j < |i|$ where i = max(i). $\text{NEXT}_M: \tilde{\Sigma}^* \to \tilde{\Sigma}^*$ be the function that maps config of M to config at next step of execution. $\forall i \in \mathbb{N}$, $\text{NEXT}_M(\alpha)_i$ only depends on the coordinates $\alpha_{i-1}, \alpha_i, \alpha_{i+1}$. TM to NAND-TM head position → index, state space → scalar vars, tape/alphabet → arr vars. NEXT_M → lines of NAND

Chap 8. Universality and Uncomputability

Universal TM U can evaluate all other machines, including machines more complex (w/ more states) than U. **Definition** $\exists$ TM U s.t. for every M representing a TM and $x \in \{0,1\}^*$, $U(M,x) = M(x)$. If M halts on x and outputs y, $U(M,x) = y$. If M doesn't halt ($= \perp$), $U(M,x) = \perp$. **TM → String** TM M has k states and size ℓ alphabet $\Sigma = \{\sigma_0 \cdots \sigma_{\ell-1}\}$. M = triple $(k, \ell, T = (\delta_M(1,0), \delta_M(0,\sigma_0) \ldots, \delta_M(k-1, \sigma_{\ell-1})))$. $\delta_M = (s', \sigma', d')$ where $s' \in [k]$, $\sigma' \in \Sigma$, $d \in \{0, 1, 2, 3\}$ $\{L, R, S, H\}$. ∴ M is encoded by 2.3$k\ell$ length list. **Uncomputable** **Uncomputable Exists** Diagonalization on inputs $\{0,1\}^*$ and $\{$programs $G_i\}$. Define $F^*(x) = 1 - G_i(x)$ $\forall x$ and i. **HALT** $\forall M \in \{0,1\}^*$, HALT$(M,x) = 1$ if TM M halts on input x and 0 o/w. HALT is uncomputable. **Rice's Theorem.** **Semantic properties**: properties of the function a program computes as opposed to properties that depend on particular syntax used by a program. We cannot certify semantic properties of general purpose programs. **Semantic Functions** Definition 8.14, Thm M and M' are functionally equivalent if for every $x \in \{0,1\}^*$, $M(x) = M'(x)$ (or one halts iff the other does). $F: \{0,1\}^* \to \{0,1\}$ is semantic if for every pair of strings M, M' that represent = TMs, $F(M) = F(M')$. **Rice's Thm** Thm 9.15 Let $F: \{0,1\}^* \to \{0,1\}$. If F is semantic and nontrivial, it is uncomputable. **Chap 10. Is every thm provable?** **Mathematical Statements** Assert properties of any mathematical set, including sets, strings, functions, graphs, and programs. **Proof System** Definition 10.2 A proof for a statement $X: \{0,1\}^*$ is another piece of text $w \in \{0,1\}^*$ that certifies the truth of the statement asserted in X. (conditions: (1) **Effectiveness**: $\forall x, w \in \{0,1\}^*$, $V(x,w)$ halts w/ output 0 or 1. Aka an algorithm exists for verification. (2) **Soundness** $\forall x \notin T$ and $w \in \{0,1\}^*$, $V(x,w) = 0$, aka there is a valid proof w for x then x is true. A true statement $x \in T$ is unprovable if for every $w \in \{0,1\}^*$, $V(x,w) = 0$. **Complete** V is complete if there $\nexists$ a x that is unprovable with respect to V.

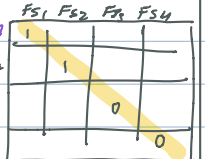
Gödel's Incompleteness Thm. Thm 10.3 There does not exist a complete proof system for H (set of strings representing TMs! $x \in \{0,1\}^*$ that have the form "TM M halts on the zero input." → $\forall$ sound proof system V for sufficiently rich math statements, $\exists$ statement that is true but not provable in V.

TM Machine



Eat your cake and Have it too inj: one-to-one, surj: onto
- prove something can't be done w/ simple TM model
- prove something can be done w/ feature-rich RAM Machine

Cardinality of Inf functions: Diagonalization



Each F_{S_i} is defined on each string $\{0,1\}^*$ and the cells in the table represent the output of F_{S_i}. We can construct a $G(S_i)$ = opposite of $F_{S_i}(S_i)$. G will differ from the output of F_{S_i} on the i^{th} string. ∴ not onto because there'll some G not in mapping. No map from $\{0,1\}^* \to \{0,1\}^*$

Uncomputable: Reduction (non-semantic) We can't in all cases, $P_F(X) = F(X)$ which contradicts the fact F is uncomp.

EX uncomputable functions: HALT, $F^*(x) = \begin{cases} 1 & x(x) = 0 \\ 0 & \text{o/w} \end{cases}$, HALTONZERO$(M)$ = 1 if $M(0)$ halts

ZEROFUNC: $\forall M \in \{0,1\}^*$ ZEROFUNC$(M) = 1$ iff M always outputs 1.

Analysis can't work on $F(M,X)$ function.

Assume by contradiction that $G: \{0,1\}^* \to \{0,1\}$ is computable: P_G computes G. If $P_G \exists$, can use P_G to compute uncomputable F. Reducing F to G.

```
def P_F(M, X):
    transform X_F to X_G
    return P_G(X_G)   ← return P_G(N)

def N(w):   ← run M on X
    y ← U(M,X)
    returns 1 and halts if some condition
```

Uncomputable: Rice's Theorem (semantic)

Semantic: Consider two functionally eq TMs M, M'. Case work for outputs of G to show $M(x) = 0 = M'(x)$ or $M(x) = 1 = M'(x)$ always. In all cases, $G(M) = G(M')$, so G is semantic.

Nontrivial: Give examples of output 1 and output 0 → ∴ not constant func, non-trivial

LOOKUP show every finite function is computable by NAND-CIRC → LOOKUP stores table of input & output.

Proving Computability.

Computable: proof by Python

"Eat your cake and have it too": Computable by Python → computable by TM. Explain why the python function computes function.

F not semantic ⇒ F is computable.

Semantic takes 1 TM. $EQ = \begin{cases} 1 & M \cong N \\ 0 & \text{o/w} \end{cases}$

$F(M) = EQ(M, N_0)$, $N_0(w) = XOR$ (bits of w)

$F = CX(M)$, which is semantic

Comparing Power XOR is not monotone.

XOR < NAND Proof. Induction on # lines of P. $k=1$, $F(x) = b + \sum_{i \in S \subseteq [n]} x_i \bmod 2$.

of form $y_0 = XOR(x_\alpha, x_\beta)$.

$k+1$ $F'(x) = b' + \sum_{i \in S' \subseteq [n]} x_i \bmod 2 = y_\alpha + y_\beta \bmod 2$

using $(a \bmod 2 + b \bmod 2) \bmod 2 = (a+b) \bmod 2$

$\to F'(x) = b_\alpha + b_\beta + b + \sum x_i \bmod 2$.

If XOR could compute NAND, it has same form.

NAND$(0,1)$ = NAND$(0,0)$, so $1 \notin S$. NAND$(1,0)$ = NAND$(0,0)$, so $0 \notin S = \emptyset$. F is constant 0 or 1 → contradiction ■

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

UNIFORM COMPUTATION

Ch 9. Restricted Computational Models

Regular Expressions **Def.** A regex e over an alphabet Σ is a str over $\Sigma \cup \{(,), |, *, \emptyset, ""\}$ 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. **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 $m = |e|$ of e, aka # of symbols. Most efficient matching algo is $O(n)$. **DFA** Def. DFA w/ ℓ states over $\{0,1\}$ is a pair (T,S) w/ transition func $T: [\ell] \times \{0,1\} \to [\ell]$ and set of accepting states $S \subseteq [\ell]$. (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 no s.t. $\forall w$ with $F(w) = 1$ and $|w| > n_0$, 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 $w = 0^{n_0} 1^{n_0}$ (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) choose any k. $k=2$: $xy^2z = 0^{n_0+b} 1^{n_0}$, $F(xy^2z) = 0$ → contradiction.

REG ⊆ TIME$(O(n)) \subseteq P$ ↳ closed under complement

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 $F(x)$. **TIME** $TIME_{TM}(T(n))$ = set of bool funcs computable in $T(n)$ TM time. TIME is class of unbounded input length functions. $TIME(T(n)) \subseteq SIZE(T(n)^a)$ for some $a \in \mathbb{N}$ **NAND-RAM** same except at most $T(n)$ lines. **RAM $\leftrightarrow$ TM** $n \mapsto T(n)$ can be computed by TM in $O(T(n))$. $TIME_{TM}(T(n)) \subseteq TIME_{RAM}(10 \cdot T(n)) \subseteq TIME_{TM}(T(n)^4)$ **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-colorsum, 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 $C \cdot T$ steps, where $C \leq a|P|^b$. **Timed Universal TM** (Thm 12.8) Let $TIMEDEVAL: \{0,1\}^* \to \{0,1\}^*$ be defined as $TIMEDEVAL(M,x,1^T) = \begin{cases} M(x) & M \text{ halts within} \leq T \text{ steps on } x \\ 0 & \text{o/w} \end{cases}$, then $TIMEDEVAL \in P$. **Time Hierarchy Thm.** **Def** For every nice function $T: \mathbb{N} \to \mathbb{N}$, there $\exists F: \{0,1\}^* \to \{0,1\}$ in $TIME(T(n)\log(n)) \setminus TIME(T(n))$ (any efficiently computable fnct that tends to ∞) **Corollary** $P \subsetneq EXP$. **Ex:** Bounded HALT $F(M,x) = 1$ if M halts w/ at most $|x|^{\log|x|}$ steps. **Non:** $P \subsetneq TIME(n^{\log n}) \subseteq TIME(2^n) \subseteq TIME(2^{n^2}) \subsetneq EXP \subsetneq TIME(2^{2^n})$. **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, $F \in SIZE(T(n))$ 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** $= \bigcup_{c \in \mathbb{N}} SIZE(n^c)$. **$P \subsetneq P/poly$** (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 $F: \{0,1\}^* \to \{0,1\}$, $TIME(T(n)) \subseteq SIZE(T(n)^a)$. **P/poly has uncomputable functions** $UHALT: S: \mathbb{N} \to \{0,1\}^*$ outputs binary rep of input w/o most significant 1 (onto function). $\forall x \in \{0,1\}^*$, $UHALT(x) = HALTONZERO(S(|x|))$. UHALT always = 1 or always = 0 → NAND-TM rep exists w/ const # of lines. **Unknown** Is $EXP \subseteq P/poly$? 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: $H \leq_p F$. (can solve H using a F solver, if F is easy H is easy, if H is hard F is hard) (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**: (for every x' assignment, it came from one) 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. $H(x) = 1$. **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:** (edges more important) $\{0,1\}^* \to \{0,1\}$ 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. (O(en) if $|x_i| = e$ bits) **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. $\sum_{i \in S} x_i = t$. (algo that runs in poly $(n, \log t)$) **Integer Programming** Input list of m linear inequalities in n variables $x_0 \cdots x_{n-1}$ with each inequality of the form $a_0 x_0 + a_1 x_1 + \cdots + a_{n-1} x_{n-1} \geq b$. 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 0/1 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. (vertices important) **VERTEXCOVER** Input: $G = (V,E)$ and number k. Outputs 1 if $\exists$ set S of $|S| \leq k$ s.t. $\forall$ edge (u,v), either $v \in S$ or $v \in S$. ($k_V = |V| - k_I$. S is vertex cover if V/S is ISET.) **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. **CLIQUE** Input: G, k. Output 1 if $\exists S$ of $|S| \geq k$ vertices s.t. $\forall v \neq v \in S$, $(u,v) \in E$. **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 multiple 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** $\leftrightarrow$ Eqns $\neg x_i = y_i$, add $x_i + y_i = 1$ (only one can be one). **Force variables = each other** $(\bar{a} \vee b) \wedge (\bar{b} \vee c) \wedge (\bar{c} \vee d) \wedge (\bar{d} \vee a)$

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 $x \in \{0,1\}^n$, $F(x) = 1 \iff \exists w \in \{0,1\}^{n^a}$ s.t. $V(x,w) = 1$. **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. $\overline{3SAT(\varphi)} = 1 - 3SAT(\varphi)$ 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 $\in P$ or no NPC $\in P$. If one NPC problem in $P_{/poly}$, then NPC $\in P_{/poly}$.

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 bit 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 $SHORTPROOF_V(x, 1^m) = 1$ iff $\exists w \in \{0,1\}^*$ with $|w| \leq m$ s.t. $V(xw) = 1$. If P = NP, despite Gödel's Incompleteness Thm, automate finding proofs. **Poly Hierarchy Collapse** $\forall a \in \mathbb{N}$, polynomial p, and poly-time algo $V \to \exists$ poly-time $SOLVE_{V,a}$ that on input $x \in \{0,1\}^n$ returns 1 iff $\exists y_0 \in \{0,1\}^m$ $\forall y_1 \in \{0,1\}^m \ldots Q_{y_{a-1}} \in \{0,1\}^m$ $V(xy_0 \ldots y_{a-1}) = 1$

RANDOMIZED COMPUTATION

Ch 17. Prob Theory

101 RVs $\mathbb{E}[X] = \sum_{x \in \{0,1\}^n} 2^{-n} X(x)$ **Linearity of Expectation** $\mathbb{E}[X+Y] = \mathbb{E}[X] + \mathbb{E}[Y]$ **Union Bound** for every A, B: $Pr[A \cup B] \leq Pr[A] + Pr[B]$ **Independence Def** Ind. if $Pr[A \cap B] = Pr[A] \cdot Pr[B]$. **Disjoint** separate from ind. if $A \cap B = \emptyset$. **Positively correlated** $Pr[A \cap B] > Pr[A] \cdot Pr[B]$. **Conditional Prob.** $Pr[B|A] = Pr[A \cap B] / Pr(A)$. **More disjoint** If X, Y are rvs that depend on two disjoint sets of coins S, T, then they are independent. $\mathbb{E}[\prod_{i=0}^{n-1} X_i] = \prod_{i=0}^{n-1} \mathbb{E}[X_i]$. Functions also preserve independence. **Concentration/Tail Bounds** **Markov's Inequality** If X is a non-negative rv then $Pr[X \geq k\mathbb{E}[X]] \leq 1/k$. Not tight **Chebyshev's Inequality** Suppose that $\mu = \mathbb{E}[X]$ 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 $\mathbb{E}[X_i] = p$ 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, $\bar{S}$) that cuts at least m/2 edges in expectation. **Amplification Idea.** Repeat process several times and output best cut. Use inequality $(1 - 1/k)^k \leq 1/e \leq 1/2$. 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 $Pr[A(x) = F(x)] \geq \frac{1}{2} + \frac{1}{p(n)}$ $\forall x \in \{0,1\}^*$ then for every $g(n)$ there $\exists$ poly-time algo B satisfying $Pr[B(x) = F(x)] \geq 1 - 2^{-g(n)}$ for every $x \in \{0,1\}^*$.

Ch 19. Modelling Randomized Computation

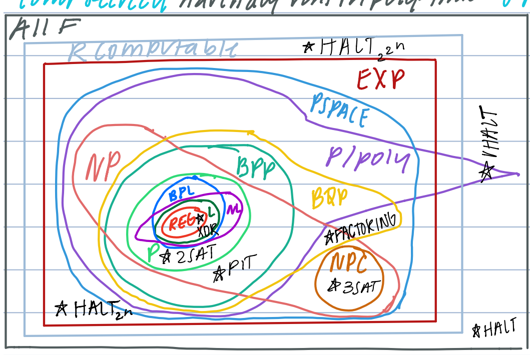
BPP Def. $F \in BPP$ if $\exists a, b \in \mathbb{N}$ and RNAND-TM program P s.t. $\forall x \in \{0,1\}^*$, on input x, P halts within at most $a|x|^b$ steps and $Pr[P(x) = F(x)] \geq 2/3$ where this prob is taken over the result of the RAND operations of P. (not over choice of input x. Must be ≥ 2/3 on every possible input.) (can amplify arbitrarily close to 1) **Alt Def.** $F \in BPP$ iff $\exists a, b \in \mathbb{N}$ 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 ineffective** 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. $\forall x$, $Pr[B(x) = F(x)] \geq 1 - 2^{-q(n)}$. ($(1 - 1/k)^k \leq 1/e \leq 1/2 \to$ fail$^{k \cdot 1000} \leq 1/e^{1000} \leq 1/2^{1000}$) **NP-hard and BPP** If F is NP-hard and $\in$ BPP, then NP $\subseteq$ BPP. (≡ if 3SAT ∈ BPP) We get essentially all P = NP consequences. Unknown if BPP = EXP. (unknown how to rule out b/c of conj. existance of pseudorand generators.) Unknown if BPP = P. (if true, supports extended Church–Turing hypothesis that deterministic poly-time algos capture feasibly computable) Suspected as. BPP ⊆ EXP. Iterate through all r inputs ($2^{a|x|^b} \cdot$ poly) and maj vote P ⊆ BPP. Throw away randomness. BPP ⊆ P/poly Yes, can show $\exists r$ s.t. it is correct for all inputs (but don't know what it is) via amplification. (if 3SAT ∉ P/poly, NP ⊄ BPP (widely believed)) Can't have P = BPP = EXP. Unknown whether BPP ⊆ NP, NP ⊆ BPP, or incomparable. (→ if NP-hard F ∈ BPP) **Sipser-Gacs Theorem.** If P = NP, then BPP = P. (If BPP ≠ P, then P ≠ NP.) **Ladner's Thm:** P ≠ NP → $\exists F \in NP \setminus P$ and F is not NPC.

Space Complexity.

Problems CIRC-EVAL, PATH. **SPACE(s(n))** = computed by TM w/ space s(n). (→ O(n)) (→ ∈ NL, NL-complete, ∈ L^2) **SPACE(O(1))** = REG **L** = $\bigcup_{c \in \mathbb{N}}$ SPACE(c log n) (feasible) **PSPACE** = $\bigcup_{c \in \mathbb{N}}$ SPACE(n^c) (infeasible, contains some NPC) (↳ F ∈ PSPACE ⇒ poly-space V exists) $L^2 = \bigcup_c SPACE(c \log^2 n)$ **Space Heirarchy Thm** $\forall s_1(n), s_2(n): \mathbb{N} \to \mathbb{N} > \log n$ s.t. $s_1(n) = o(s_2(n)) \to SPACE(s_1(n)) \subsetneq SPACE(s_2(n))$. $TIME(f(n)) \subseteq SPACE(f(n)) \subseteq TIME(2^{f(n)}) \to L \subseteq P \subseteq PSPACE \subseteq EXP$. (cannot all be equal) (L ⊆ NL ⊆ P) $L \subsetneq L^2 \subsetneq PSPACE$. **Universal TM Space** U(M, x) takes O(s(|x|) + log(|x|)) **Randomness** BPPSPACE, BPL. Algo tosses coins or access to random string. **Nondeterminism** (NPSPACE, NL) Algo takes input on input tape and witness on witness tape or two-way input tape access + one-way witness tape access. **Facts** $BPSPACE(s(n)) \subseteq SPACE(s(n)^2) \leftrightarrow BPPSPACE = PSPACE \leftrightarrow L \subseteq BPL \subseteq L^2$. $NPSPACE(s(n)) \subseteq SPACE(s(n)^2) \leftrightarrow NPSPACE = PSPACE \leftrightarrow L \subseteq NL \subseteq L^2$. NL vs BPL unknown.

Cryptography.

Definitions ciphertext: $y = E_k(x)$, Decoder: $x = D_k(y)$, plaintext $x \in \{0,1\}^*$, secret key $k \in \{0,1\}^*$. **Validity** (E, D) valid if $\forall k \in \{0,1\}^n$ $\forall x \in \{0,1\}^{L(n)}$ $D_k(E_k(x)) = x$ **Secure** if adversary cannot learn anything about plaintext. **Perfectly Secret** if for every $x, x' \in \{0,1\}^{L(n)}$, $\{E_k(x)\}_{k \sim \{0,1\}^n}$ and $\{E_k(x')\}_{k \sim \{0,1\}^n}$ are identical dists ↔ Pr[Adversary guesses whether $y = E_k(x)$ or $= E_k(x')$] ≤ 1/2 **Limitation** If (E, D) perf secret, then |key size| ≥ |message size| (n, L(n)) (identity of discernables principle ↙) **Comp secrecy** Adversary runs in poly-time **∃ Pseudorandom Gens** then ∃ comp encryption w/ secret |key| << |message| (stronger assumption than P ≠ NP) **Public/Private** public $e \in \{0,1\}^*$, ciphertext $y = E_e(x)$ (→ impossible if P = NP) generate key pair $r \sim \{0,1\}^n$ $(e, d) = G(r)$, decryption $x = D_d(y)$. **One-time-Pad** valid scheme (E, D) w/ L(n) = C(n) = n. (perf secret → x ⊕ k)



Quantum

Bell's Ineq ∃ game by prover1, prover2, verifier s.t. if P1, P2 share classical bits, Pr[win] ≤ 3/4; share quantum state, Pr[win] ≥ 0.85 $P_{/poly} \subseteq BQP_{/poly}$ (comp by NAND ≤ quant)

≤	P	NP	NPC	P/poly	T(2^{n^2})	BPP	REG	R	BQP	EXP
P	▨	≤	⊄	⊊	⊊	≤	⊄	⊊	≤	⊊
NP	?	▨	⊄	?	?	?	⊄	⊊	?	≤
NPC	?	⊊	▨	?	?	?	⊄	⊊	?	⊊
P/poly	⊄	⊄	⊄	▨	⊄	⊄	⊄	⊄	⊄	⊄
T(2^{n^2})	⊄	?	⊄	?	▨	?	⊄	⊊	?	≤
BPP	?	?	⊄	⊊	≤	▨	⊄	⊊	?	≤
REG	⊊	⊊	⊊	⊊	⊊	⊊	▨	⊊	⊊	⊊
R	⊄	⊄	⊄	⊄	⊄	⊄	⊄	▨	⊄	⊄
BQP	?	?	⊄	?	?	?	⊄	⊊	▨	≤
EXP	⊄	?	⊄	?	?	?	⊄	⊊	?	▨

⊊ · ≤ · ⊄ · ?

BPP = P · P = NP · BQP = PSPACE · BPP = EXP · P = BPP = EXP

CoNP class of negations of NP problems (→ ex: UN3SAT) (coNP ≠ NP stronger than P ≠ NP) P = NP → NP = coNP. P = NP → NP ⊆ P/poly (∈ ε)

Pseudorandom Gens (every C w/ m input, T gates) $|Pr_{s \sim \{0,1\}^\ell}[C(G(s)) = 1] - Pr_{r \sim \{0,1\}^m}[C(r) = 1]|$ **Optimal PRG conjecture** exponentially secure PG exists → BPP = P (reduces to m coin → O(log m) algo)