Undergraduate Complexity Theory **CMU 15-455, Spring 2017**

Some Final Practice Problems

No warranty is made or implied regarding whether these are good problems, nor whether they are harder, easier, or about the same difficulty level as the problems on the midterm, nor whether they achieve good coverage of the topics on the final exam.

1. Write the definition of the following terms:
   - alphabet
   - string
   - $\langle X \rangle_\Sigma$
   - decision problem
   - function problem
   - search problem
   - language
   - Boolean circuit
   - Boolean formula
   - Boolean function
   - the PATH (also known as ST-PATH) problem
   - the PALINDROMES problem
   - the BOUNDED-ACCEPTANCE$_{f(n)}$ problem
   - the kCOL problem (for $k \geq 2$)
   - the LCS (longest common subsequence) problem
   - the CLIQUE and k-CLIQUE problems
   - the HAMILTONIAN-PATH problem
   - CIRCUIT-SAT, (FORMULA-)SAT, CNF-SAT, kSAT, EkSAT, and NAEkSAT problems
   - the CIRCUIT-EVAL problem
   - CNF formula, DNF formula, literal
   - Church–Turing Thesis
   - Extended Church–Turing Thesis
   - Turing Machine
   - transition function
   - configuration
   - computation trace of a Turing Machine
   - decider
   - Turing Machine M decides language L
   - Turing Machine M runs in time $f(n)$

- $f(n)$ is $O(g(n))$
- multitape Turing Machine
- nondeterministic pseudocode / Turing Machines (including what it means for such a machine to "accept string x" and what its "running time" is)
- universal Turing Machine
- time-constructible function, Time Hierarchy Theorem
- $\mathsf{TIME}(f(n))$
- $\mathsf{NTIME}(f(n))$
- P
- NP
- EXP
- NEXP
- V is a verifier for language L
- polynomial-time verifier
- Exponential Time Hypothesis (ETH), Strong Exponential Time Hypothesis (SETH)
- polynomial-time mapping reductions ($A \leq_m^P B$)
- NP-hard
- NP-complete
- search-to-decision reduction
- "padding" (we didn't give a completely formal definition, but give the concept)
- Ladner's Theorem
- Mahaney's Theorem
- coNP, $\mathsf{NP} \cap \mathsf{coNP}$, and the prefix co- in general
- the UNSAT, TAUT problems
- co-nondeterministic algorithm
- Turing Machine M runs in space $f(n)$
- read-only input tape, read-once input tape, work tape, write-once output tape
- $\mathsf{SPACE}(f(n))$
- L, NL, PSPACE
- (the idea of) log-space pseudocode
- configuration of a (space-bounded) TM with read-only input tape
- space-constructible function, Space Hierarchy Theorem
- Savitch's Theorem
- $\mathsf{NSPACE}(f(n))$
- configuration graph of a (non)deterministic space-bounded TM computation
- $A \leq_m^L B$
- A is $\mathcal{C}$-complete under log-space reductions (where $\mathcal{C}$ is a complexity class)
- the TQBF problem (also known as QSAT)

- Immerman–Szelepcsényi Theorem
- a probabilistic Turing Machine, M
- M decides language A with one-sided (resp., two-sided) error ε
- $\mathsf{RTIME}(f(n))$
- RP, coRP, ZPP, BPP
- $\Sigma_i\mathsf{P}$, $\Pi_i\mathsf{P}$, for $i \in \mathbb{N}$, and PH
- the polynomial hierarchy collapses
- the MIN-CIRCUIT problem
- SAT-oracle Turing Machine; more generally, A-oracle TM, where A is a language
- P^A and P^{NP}
- $A \leq_T^P B$

2. Draw a Venn diagram illustrating our knowledge of the relationship between the following complexity classes:

   BPP, E, EXP, L, NEXP, coNEXP, NL, coNL, NP, coNP, $\mathsf{NP} \cap \mathsf{coNP}$, P, PH, P^{NP}, PSPACE, $\mathsf{NPSPACE}$, RP, coRP, $\Sigma_2\mathsf{P}$, $\Sigma_3\mathsf{P}$, $\Pi_2\mathsf{P}$, $\Pi_3\mathsf{P}$, $\mathsf{SPACE}(\log^2 n)$, ZPP

   Try to place a complete problem within each of the regions, if you can.

3. Prove that $\mathsf{P} = \mathsf{L} \implies \mathsf{EXP} = \mathsf{PSPACE}$. (Hint: padding.)

4. Prove $\mathsf{E} \neq \mathsf{EXP}$.

5. Prove the Space Hierarchy Theorem.

6. We say a language A is *downward self-reducible* if there is a polynomial-time oracle Turing machine that can decide whether $x \in A$ given the ability using oracle queries to A itself, *but only on strings of length strictly shorter than* $|x|$. Show two things: (a) SAT is downward self-reducible; (b) any self-reducible language is in PSPACE.

7. Prove that BPP is closed under $\leq_m^P$.

8. Define BPP^A to be the class of all languages L such that there is a probabilistic polynomial-time A-oracle Turing Machine M^A with the property that

$$x \in L \implies \mathbf{Pr}[M^A(x) \text{ accepts}] \geq 2/3,$$
$$x \notin L \implies \mathbf{Pr}[M^A(x) \text{ accepts}] \leq 1/3.$$

   Suppose $A \in \mathsf{BPP}$ and $L \in \mathsf{BPP}^A$. Show that $L \in \mathsf{BPP}$. (This result is sometimes stated as $\mathsf{BPP}^{\mathsf{BPP}} = \mathsf{BPP}$. Hint: you will need the fact that in BPP, error probabilities can be reduced to $2^{-p(n)}$ for any polynomial $p(n)$, at the cost of a polynomial slowdown.)

9. Show also that $\mathsf{ZPP}^{\mathsf{ZPP}} = \mathsf{ZPP}$. What seems to go wrong if you try to show that $\mathsf{RP}^{\mathsf{RP}} = \mathsf{RP}$?

10. Show that the language

$$\text{MOD} = \{\langle a, b, c\rangle : a, b, c \in \mathbb{N} \text{ such that } a \bmod b = c\}$$

    is in L, assuming as usual that a, b, c are encoded in binary.

11. Show that the language

$$\text{SUM-PAL} = \{\langle a, b\rangle : a, b \in \mathbb{N} \text{ such that } a + b \text{ is a palindrome}\}$$

is in L, where we say that an integer is a palindrome if its representation as a binary string is a palindrome.

12. Prove that 2SAT is NL-complete.

13. (Problem taken from Chris Umans.) A function $f : \Sigma^* \to \Sigma^*$ is called *length-preserving* if $|f(x)| = |x|$ for all $x \in \Sigma^*$. A length-preserving function f is called a *permutation* if it is a surjection, and notice that in this case it is also an injection, and hence the inverse function f^{-1} is well defined. Informally, such an f is called a *one-way permutation* if f is computable in polynomial time but f^{-1} is hard to compute. Much of cryptography is based on the assumption that there exist length-preserving, one-way permutations; indeed, with a little massaging, the multiplication of two primes can be expressed in this way, with the inverse problem being factoring.

    Let $f : \Sigma^* \to \Sigma^*$ be a length-preserving, one-way permutation, and let $g : \Sigma^* \to \{0, 1\}$ be a polynomial-time computable predicate. A typical scenario is this: you hold a secret x, and you reveal $y = f(x)$ (which is easy to compute). If an adversary can obtain x from y, she is able to compute a bit $g(x)$ that she is not supposed to know. In this problem you will show that the adversary's task is in $\mathsf{NP} \cap \mathsf{coNP}$. More precisely, show that the following language is in $\mathsf{NP} \cap \mathsf{coNP}$:

$$L = \{y : g(f^{-1}(y)) = 1\}.$$

    Remark: Since it is possible that $\mathsf{P} = \mathsf{NP} \cap \mathsf{coNP}$ yet $\mathsf{P} \neq \mathsf{NP}$, this shows that $\mathsf{P} \neq \mathsf{NP}$ is not a sufficient assumption to enable much of cryptography.

14. Consider the language L of all true quantified Boolean formulas of the form $\exists u \forall v \phi(u, v)$, where ϕ is a general Boolean formula in variables $u_1, \ldots, u_n, v_1, \ldots, v_m$. Consider also the same language L' but where ϕ must be a CNF. Using the (true) fact that L is $\Sigma_2\mathsf{P}$-complete, show that L' is $\Sigma_2\mathsf{P}$-complete.

15. Let $\phi(x_1, \ldots, x_n)$ be a Boolean formula. Consider the following two-player game: First, Alice picks a truth value for x_1; then Bob picks a truth value for x_2; then Alice picks a truth value for x_3, etc., until a truth value is picked for x_n (by Alice if n is odd, by Bob if n is even). Alice wins if the final truth value of ϕ is True, and Bob wins if the final truth value of ϕ is False. Suppose we now ask a decision problem: Given a Boolean formula $\phi(x_1, \ldots, x_n)$, does Alice have a winning strategy? Show that this decision problem is PSPACE-complete.

16. Show that the language

$$B = \{0\#1\#10\#11\#100\#101\#110\#\cdots\#\text{bin}(k) : k \in \mathbb{N}\} \subseteq \{0, 1\#\}^*$$

is in $\mathsf{SPACE}(\log \log n)$.

17. (Directly related to Lecture 28, which is not "on the final", but you still may enjoy it.) Design languages C and D such that $\mathsf{P}^C = \mathsf{BPP}^C$ and $\mathsf{P}^D \neq \mathsf{BPP}^D$.