6.045 Midterm Solutions

May 18, 2011

True/False/Open

(a) F - $1 = o(2)$.
(b) T - $n^{1/10} = \Omega(2^{\sqrt{\log n}})$.
(c) T - $4n + 2 = \Theta(n^{3/2})$.
(d) F - $\text{TIME}(2^{\sqrt{n}}) = \text{TIME}(2^n)$.
(e) T - Either $P \neq \text{NP}$ or $\text{NP} \neq \text{BQP}$ or $\text{BQP} \neq \text{EXP}$.
(f) F - If $L_1$ and $L_2$ are context-free, then $L_1 \cap L_2$ is context-free.
(g) F - Every context-free language is recognizable by some DPDA.
(h) T - There exists a language $L$ such that neither $L$ nor $\text{HALT}$ is Turing-reducible to the other.
(i) F - There exists a recognizable language that is not Turing-reducible to $\text{HALT}$.
(j) T - $\{\langle M \rangle : \exists x \in \{0,1\}^* \text{ such that } M(x) \text{ halts} \}$ is recognizable.
(k) F - $\{\langle M \rangle : \forall x \in \{0,1\}^* M(x) \text{ halts} \}$ is recognizable.
(l) T - $\{\langle M \rangle : M \text{ halts for all possible initial settings of its infinite input tape} \}$ is recognizable.
(m) F - For every function $f$, there’s a function $g$ computable with $\text{HALT}$ oracle such that $f(n) = O(g(n))$.
(n) T - There’s an $n$-qubit quantum state that requires $\Omega(c^n)$ two-qubit gates to prepare,1 for some $c > 1$.
(o) F - Every consistent axiomatic theory is sound.
(p) T - Every sound axiomatic theory is consistent.
(q) T - If $\text{ZF}$ is consistent, then there’s no computable function $f$ such that every $\text{ZF}$ theorem with $\leq n$ symbols has a $\text{ZF}$ proof with $\leq f(n)$ symbols.
(r) F - If $\text{ZF}$ is consistent, then $\text{ZF} + \text{Not}(G(\text{ZF}))$ (where $G(\text{ZF})$ is $\text{ZF}$’s Gödel sentence) is inconsistent.
(s) T - The CircuitSAT problem, for circuits of AND and XOR gates,2 is $\text{NP}$-complete.
(t) O - The CircuitSAT problem, for circuits of AND and OR gates, is $\text{NP}$-complete.
(u) O - If $\text{P} \neq \text{NP}$, then there exists a one-way function.
(v) T - If RSA is secure against polynomial-time classical adversaries, then $\text{BPP} \neq \text{BQP}$.
(w) O - Quantum computers can break any public-key cryptosystem in polynomial time.

1Starting from the initial state $|0\cdots0\rangle$
2For this statement and the next, assume the constants 0 and 1 are available for free.
(x)  O - Quantum computers can factor integers exponentially faster than classical computers can.

(y)  T - If \( P = \text{PSPACE} \) then \( \text{BPP} = \text{BQP} \).

(z)  F - There exists an oracle \( A \) such that \( \text{PSPACE}^A = \text{EXPSPACE}^A \).

(α)  O - If \( P \neq \text{NP} \) then \( \text{NP} \neq \text{coNP} \).

(β)  T - If \( \text{RP} \neq \text{coRP} \) then \( P \neq \text{RP} \).

(γ)  O - There exists an \( \text{NP} \) language that’s neither in \( P \) nor \( \text{NP} \)-complete.

(δ)  F - If there’s a polynomial-time Turing machine that decides the language \( L \) for at least 90% of inputs \( x \in \{0,1\}^n \), for all \( n \), then \( L \in \text{BPP} \).

1. Regular and Context-Free Languages

(a)  The set of all strings in \( \{0,1,\#\}^* \) containing an odd number of 1’s that are preceded by odd numbers of \#'s

(b)  The set of strings of the form \( x_1\#x_2\#\cdots\#x_{k-1}\#x_k \), where each \( x_i \) is in \( \{0,1\}^* \) and at least one \( x_i \) has an even number of 1’s

(c)  The set of strings of the form \( x_1\#x_2\#\cdots\#x_{k-1}\#x_k \), where each \( x_i \) is in \( 1^* \) and at least two \( x_i \)'s have the same number of 1’s

2. Time-Bounded Kolmogorov Complexity

(a)  An \( \text{NP} \) witness is just the Python program that outputs \( x \) in \( \leq T \) steps. Checking that the program works clearly takes time polynomial in the input length \( n + T \).

(b)  A simple counting argument: there are \( 2^n \) strings of length \( n \), but only \( \sum_{0 \leq i < n/2} 2^i < 2^{n/2} \) of them can be output by a program with less than \( n/2 \) bits (since each program outputs at most one string).

(c)  Let \( f \) be a PRG stretching (say) \( n/3 \) bits to \( n \) bits, which is computable in \( p(n) \) time. Then to distinguish \( f(x) \) from a truly random string, it suffices to decide whether \( f(x) \in L \) (which can be done in \( P \) by assumption). For all sufficiently large \( n \) we’ll have \( K_{p(n)}(f(x)) \leq n/2 \), whereas if \( y \in \{0,1\}^n \) is uniformly random, then \( K_{p(n)}(y) > n/2 \) with overwhelming probability by part (b).

3. PP

(a)  Just create a new machine \( M' \) that accepts whenever \( M \) rejects and vice versa.

(b)  A \( \text{PSPACE} \) machine can simply loop over all random strings \( r \), reusing the same memory for each one, and count the number of \( r \)'s that cause \( M(x) \) to accept.

(c)  The PP machine does the following: with probability \( \frac{1}{2} - \varepsilon \) (for some extremely small \( \varepsilon \)) it does nothing, halts and accepts. Otherwise, it picks a witness string \( w \in \{0,1\}^{p(n)} \) for the \( \text{NP} \) problem uniformly at random, and accepts if and only \( w \) is satisfying. If we set (say) \( \varepsilon = 2^{-2p(n)} \), then this gives a machine that accepts with probability \( < 1/2 \) if there are no satisfying witnesses, and with probability \( > 1/2 \) if there’s at least one satisfying witness.

(d)  An example of such a problem is \( \text{MAJSAT} \): given a \( \text{SAT} \) instance \( \varphi(x_1,\ldots,x_n) \), do at least half of the \( 2^n \) possible assignments make \( \varphi \) evaluate to \( \text{TRUE} \)?
4. Witness of Failure

Intuitively, we construct a 3SAT instance $X$ that encodes the statement: “there exists a 3SAT instance $Y$ of size $n$ on which $A$ fails.” In more detail: “either there exists a 3SAT instance $Y$ on which $A$ outputs ‘NO,’ together with a satisfying assignment for $Y$, or else there exists a 3SAT instance $Y$ on which $A$ outputs an assignment that is not satisfying.” This is an NP statement (if it’s true, then there’s an easily-checkable witness); therefore, by the Cook-Levin Theorem, it must be possible to construct a 3SAT instance $X$ that encodes the statement. Furthermore, $X$ must be satisfiable, by the assumption that $A$ does fail on some 3SAT instance of size $n$.

Now we feed $X$ to $A$ and see what happens. There are two possibilities: if $A$ finds a satisfying assignment to $X$, then it has helpfully given us a 3SAT instance $Y$ of size $n$ on which $A$ fails. On the other hand, if $A$ fails to find a satisfying assignment to $X$, then $X$ itself is the 3SAT instance on which $A$ fails that we were looking for!

5. Zero-Knowledge

The protocol is the following: assuming $G$ and $H$ are isomorphic, Merlin picks a third graph $K$ that’s isomorphic to both of them, by randomly permuting the vertices of $G$ (or alternatively, of $H$). Then Merlin sends $K$ to Arthur. Next Arthur sends Merlin a random challenge: with probability 1/2, he asks to see the isomorphism between $G$ and $K$, and with probability 1/2, he asks to see the isomorphism between $H$ and $K$. Merlin responds with the requested isomorphism, and Arthur checks that it works (and accepts if and only if it does). To boost his confidence, Arthur can repeat the protocol as many times as desired, with a different $K$ each time.

This protocol satisfies completeness because if $G \cong H$, then Merlin can clearly always give an isomorphism between $G$ and $K$ or between $H$ and $K$, and therefore Arthur accepts with probability 1. It satisfies soundness because if $G$ is not isomorphic to $H$, then with probability 1/2, Arthur will ask to see an isomorphism between $K$ and some graph to which $K$ is not isomorphic, and Merlin won’t be able to give the isomorphism (because there isn’t one). Finally, the protocol satisfies zero-knowledge because Arthur doesn’t “learn” anything he didn’t already know: he just sees either a random graph isomorphic to $G$ or else a random graph isomorphic to $H$, either of which he could have easily generated on his own (without Merlin).

6. Cryptography

Given $y \in Z_N$, our goal is to find an $x \in Z_N$ such that $x^3 = y \pmod{N}$. To do so, we can first pick an $r \in Z_N$ uniformly at random, and set $y' := r^3y$. We can then use our algorithm $M$ to search for an $x'$ such that $(x')^3 = y' \pmod{N}$. Now, as we vary $r$, notice that $y'$ is just a uniformly random element of $Z_N$. Therefore, by assumption, $M$ succeeds at finding the cube root $x'$ of $y'$ with probability at least $c$ over the choice of $r$. Furthermore, assuming $M$ succeeds, we can then set $x := x'/r$, so that

$$x^3 = \frac{(x')^3}{r^3} = \frac{y'}{r^3} = \frac{r^3y}{r^3} = y \pmod{N},$$

which is what we wanted. If $M$ doesn’t succeed, then we can just try over and over with different values of $r$ until we find one that works.

7. Quantum Teleportation

(a) $\frac{1}{\sqrt{2}} [\alpha \ket{000} + \alpha \ket{011} + \beta \ket{110} + \beta \ket{101}]$

(b) $\frac{1}{2} [\alpha \ket{000} + \alpha \ket{100} + \alpha \ket{011} + \alpha \ket{111} + \beta \ket{001} + \beta \ket{010} - \beta \ket{110} + \beta \ket{101}]$

(c) She sees $\ket{11}$ with probability $\frac{1}{4}$. Conditioned on her seeing $\ket{11}$, Bob’s state is $-\beta \ket{0} + \alpha \ket{1}$.

(d) $\begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$ converts $-\beta \ket{0} + \alpha \ket{1}$ to $\alpha \ket{0} + \beta \ket{1}$.
(e) By the No-Communication Theorem, nothing that Alice does to her qubits alone can affect the state of Bob’s qubit.

(f) When Bob measures the state $|\psi\rangle$, he doesn’t get to see the actual amplitudes $\alpha$ and $\beta$ themselves. Instead, the state collapses, leaving him with just a single bit $|0\rangle$ or $|1\rangle$.

(g) When Alice measures the $X$ qubit to figure out which classical message to send Bob, she destroys her copy of $|\psi\rangle$. (Furthermore, we know that there can be no clever way to avoid this, because of ... the No-Cloning Theorem!)

8. Turing Machines with Advice

(a) An uncountable number. Every possible unary language $L \subseteq \{1, 11, 111, \ldots\}$ (of which there’s an uncountable infinity) is in $P/poly$, since the $n^{th}$ advice string $a_n$ can simply “hardwire” whether or not $1^n \in L$.

(b) Let $M_1, M_2, \ldots$ be a standard enumeration of Turing machines. Then $L = \{1^n : M_n(\ ) \text{ halts}\}$ is clearly undecidable, but is in $P/poly$, again because the $n^{th}$ advice string $a_n$ can “hardwire” whether or not $1^n \in L$.

(c) By the union bound,

$$\Pr_r [A(x, r) \text{ outputs the wrong answer for some } x \in \{0, 1\}^n] < \frac{2^n}{2^n} = 1.$$ 

Hence, the probability over $r$ that $A(x, r)$ outputs the right answer for every $x \in \{0, 1\}^n$ is greater than zero. This means, in particular, that there exists at least one $r = r_n$ with this property.

(d) To simulate a BPP machine in $P/poly$, we simply set $a_n := r_n$, where $r_n$ is the string from part (c) that allowed $A(x, r_n)$ to output the right answer for every input $x \in \{0, 1\}^n$.

(e) Yes, the containment is strict, because $P/poly$ contains uncomputable problems by parts (a) and (b) (whereas BPP clearly doesn’t)!