MIRI. Computational Complexity. Final exam.

Last name:
Given name:

- Date: Thursday, June 6, 2019
- Time: 11:30 to 13:30
- Grades will be published on Friday, June 7, 2019
- There are eight multiple-choice questions
- Each question is worth 1 point
- Each question has four choices and exactly one correct answer
- A choice is considered correct only if all the statements in it are correct
- An incorrect answer does **NOT** penalize
- Circle your choice in the same sheet
- **DO NOT** unstaple the sheets
- $\exp(n)$ means 2^{cn} for some unspecified constant $c > 0$
- $\mathrm{poly}(n)$ means n^c for some unspecified constant $c > 0$
- $\log(n)$ means $c \log_2(n)$ for some unspecified constant $c > 0$

Question 1 The Time-Hierarchy Theorem, as we discussed it in class, states that:

1. DTIME(f) $\subsetneq$ DTIME(g) if f is time-constructible and $f = o(g)$.
2. DTIME(f) $\subsetneq$ DTIME(g) if f is time-constructible and $f^2 = o(g)$.
3. DTIME(f) $\subsetneq$ DTIME(g) if f is time-constructible and $f = o(g^2)$.
4. DTIME(f) $\subsetneq$ DTIME(g) if f is time-constructible and $f^2 = O(g)$.

Question 2 Professor X claims that we are currently able to prove that P $\neq$ EXP but that we do not know how to apply the same method to prove that P $\neq$ NP. Only one of the following statements is correct. Which one?

1. This is not true; we do not even know if P $\neq$ EXP.
2. This is true; but the problem is not that we do not know how to apply the same method; the problem is that NP has complete problems while EXP does not.
3. This is true; in DTIME(2^n) we are able to simulate every deterministic poly-time computation for sufficiently large inputs, and hence diagonalize against it by *doing the opposite*, while in NTIME(n^2) we can still simulate every deterministic poly-time computation for sufficiently large inputs but we do not know how to *do the opposite* once the simulation is over.
4. This is true; in DTIME(2^n) we are able to simulate every deterministic poly-time computation for sufficiently large inputs, and hence diagonalize against it by *doing the opposite*, but we do not know how to, in NTIME(n^c) for any fixed constant $c > 0$, simulate every deterministic poly-time computation for sufficiently large inputs.

Question 3 The proof of Cook's Theorem that we gave in class:

1. Required three or four full lectures to teach.
2. Fits in a blackboard.
3. We didn't give this in class: it's outside the scope of the course.
4. Is radically different from what Cook did in 1971.

Question 4 Recall that a Diophantine equation is a polynomial equation of the type $P(x_1, \ldots, x_n) = 0$ that has integer coefficients, where $x_1, \ldots, x_n$ are variables ranging over the integers. Only one of the following statements is true about the problem of checking if a given Diophantine equation, with all coefficients and exponents written in binary, has an integer solution:

1. The problem is in NP because if we are given a candidate solution with integers written in binary, then we can check that it is a correct solution in polynomial time because addition and multiplication of integers can be executed in polynomial time when numbers are written in binary.
2. The first answer is wrong because, among other things, there is no computable bound on the size of the smallest solution (in terms of the number of bits that it takes to write it down) and hence the candidate solution does not make a valid certificate according to the definition of NP.
3. The first answer is wrong but the second also: although there *is* a polynomial bound on the size of the smallest solution, the problem with the first answer is that the degree of P could be so large that the intermediate values that appear during the evaluation of the polynomial could have exponential size even if written in binary.
4. All three answers above are wrong.

Question 5 If P = NP, then

1. the polynomial-time hierarchy PH collapses to P,
2. it could well be that the polynomial-time hierarchy PH does not collapse,
3. NP = co-NP but $\Sigma_2\mathrm{P} \neq \Pi_2\mathrm{P}$,
4. the polynomial-time hierarchy PH collapses to $\Sigma_2\mathrm{P} \cap \Pi_2\mathrm{P}$ but it is not known whether it collapses all the way down to P.

Question 6 Let $A = \{x \in \{0,1\}^* : \exists y \in \{0,1\}^{p(|x|)} \text{ s.t. } \langle x, y\rangle \in B\}$ be an NP problem, i.e., B is the polynomial-time computable verifier and $p(|x|)$ is the polynomial length of the witnesses. For an $x \in A$, let $S_A(x) = \{y \in \{0,1\}^{p(|x|)} :$

$\langle x, y\rangle \in B\}$ be the set of witnesses. At a high level, the main idea for approximately counting the number of witnesses $|S(x)|$ of a YES instance x is the following:

1. Since the number of witnesses is bounded by a polynomial, there is always an efficiently computable hash function that is injective on the set of witnesses: the method for approximately counting the number of witnesses exploits this.
2. Since we are happy with an approximation, it suffices to be able to tell the following two cases apart: 1) $|S(x)^k| \geq 2^{t+c}$ vs. 2) $|S(x)^k| \leq 2^{t-c}$, where c and k are fixed constants and $t = \lfloor \log_2(|S(x)^k|) \rfloor$. The method exploits this by noting that a random hash function with range $\{0,1\}^t$, when evaluated on $S(x)^k$, is likely to hit any fixed output in case 1), and is unlikely to do so in case 2).
3. Since we are happy with an approximation, it suffices to hash the witnesses to sufficiently few bits, even if this incurs an information loss, because then the number of hashes is polynomial in the size of the input and is hence efficiently computable.
4. All this is crap: one cannot approximately count the number of witnesses in polynomial time, in full generality, even with access to an oracle in the polynomial hierarchy, unless P = NP.

Question 7 In class we proved that the complement of the Polynomial Identity Testing (PIT) problem for arithmetic formulas is in RP as an application of the Schwartz-Zippel Lemma: Given E and F, sample numbers $a_1, \ldots, a_n$ uniformly and independently at random from $\{1, \ldots, N\}$, where $N = \mathrm{poly}(\mathrm{size}(E) + \mathrm{size}(F))$, and check that $E(a_1, \ldots, a_n) \neq F(a_1, \ldots, a_n)$.

1. If we chose $N = \log(\mathrm{size}(E) + \mathrm{size}(F))$, then the analysis of the algorithm via the Schwartz-Zippel Lemma would give an error probability that is still bounded by a constant smaller than $1/2$.
2. If we chose $N = \exp(\mathrm{size}(E) + \mathrm{size}(F))$, then the analysis of the algorithm via the Schwartz-Zippel Lemma would give an error probability as small as inverse exponential in the sizes of E and F but it would not be possible to implement the algorithm so that it still runs in time $\mathrm{poly}(\mathrm{size}(E) + \mathrm{size}(F))$.

3. If we chose $N = \exp(\text{size}(E) + \text{size}(F))$, then the analysis of the algorithm via the Schwartz-Zippel Lemma would give an error probability as small as inverse exponential in the sizes of E and F, and the algorithm could still be implemented to run in $\text{poly}(\text{size}(E) + \text{size}(F))$.
4. All of the above are incorrect.

Question 8 Consider the following decision problem: Given a Boolean circuit $C(x_1, \ldots, x_n, y_1, \ldots, y_n)$ with $2n$ inputs, determine if the graph $G_C = (V_C, E_C)$ specified below is connected:

$$V_C = \{0,1\}^n,$$
$$E_C = \{\{(a_1, \ldots, a_n), (b_1, \ldots, b_n)\} : C(a_1, \ldots, a_n, b_1, \ldots, b_n) = 1\}.$$

Which one is correct?

1. The problem is in P since we can check for connectivity of the graph G_C in poly-time by a straightforward depth-first search.
2. The problem is in NPSPACE but it is not known if it is in PSPACE.
3. The problem is in EXP but it is not known if it is in PSPACE.
4. The problem is in PSPACE.
