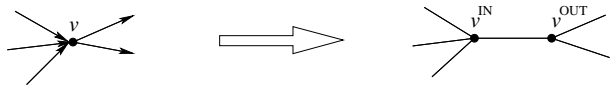
MIRI. Computational Complexity. Final exam.

- Date: Thursday, June 17, 2020. Time: 12:00 to 15:00
- Grades will be published on Thursday, June 24, 2020
- There are twelve multiple-choice questions Q1,...,Q12
- Each question is worth 5/6 points (total = 10)
- 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
- Handwrite your answers as a string $x = x_1 \cdots x_{12} \in [4]^{12}$ on white paper
- Handwrite one-paragraph justification for **Q2**, **Q5**, and **Q8**
- Scan your solutions (CamScanner, Drive, etc) and upload the pdf to the raco
- Late submission by other means cannot be accepted
- You may use textbooks, notes, or internet, if you find it useful
- You **must not** comunicate with anyone (except me via G-meet) before 15:00
- Do **not** text anyone or anywhere until after 15:00
- $\exp(n)$ 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 Savitch's Theorem states that:

1. NTIME(f) $\subseteq$ TIME(f^2),
2. NSPACE(f) $\subseteq$ SPACE(f^2),
3. NSPACE(f) $\supseteq$ SPACE(f^2),
4. NTIME(f) $\subseteq$ SPACE(f^2).

Question 2 In the Directed Hamiltonian Graph problem (DHAM) we are given a directed graph G and we are asked if G has a cycle that goes through each vertex exactly once. The Undirected Hamiltonian Cycle problem (UHAM) is the same but for ... guess what ... undirected graphs. Alice says "Here is what you do to reduce DHAM to UHAM: replace each vertex v of the directed graph as follows:"



1. This is not correct, some YES instances of DHAM get mapped to NO instances of UHAM.
2. This is not correct, some NO instances of DHAM get mapped to YES instances of UHAM.
3. On some directed graphs the transformation is not well-defined.
4. This is correct.

Question 3 The problem k-NAE-SAT is a variant of k-SAT in which the constraints are not disjunctions of literals but NAE constraints: a NAE-constraint is a k-tuple of variables $(x_{i_1}, \ldots, x_{i_k})$ that is satisfied by a $0/1$-assignment to the variables if and only if not all variables get the same value (NAE means *Not All Equal*; not all 0 and not all 1). The problem is stated as: Given a list of NAE-constraints on n variables, is there a $0/1$-assignment to the variables that satisfies all the constraints?

1. This is a well-known NP-complete problem even for $k = 2$.

2. This problem is not even in NP because every constraint has $2^k - 2$ satisfying assignments; exponentially many.

3. This problem is in NL for every $k \geq 2$ by easy reduction to 2-SAT and the fact that NL = co-NL.

4. None of the above is correct.

Question 4 If $\Sigma_{i+1}^{\mathrm{P}} = \Sigma_i^{\mathrm{P}}$ for some $i \geq 1$, then:

1. The polynomial-time hierarchy PH collapses to P.
2. The polynomial-time hierarchy PH collapses to $\Sigma_i^{\mathrm{P}} = \Pi_i^{\mathrm{P}} = \mathrm{PH}$.
3. The polynomial-time hierarchy PH collapses to $\Sigma_i^{\mathrm{P}} = \mathrm{PH}$ but it could still be that $\Sigma_i^{\mathrm{P}} \neq \Pi_i^{\mathrm{P}}$.
4. The polynomial-time hierarchy PH collapses *from below*; i.e., $\mathrm{P} = \Sigma_j^{\mathrm{P}} = \Pi_j^{\mathrm{P}}$ for all $j \leq i + 1$ but it could still be that $\Sigma_k^{\mathrm{P}} \neq \Pi_k^{\mathrm{P}}$ for all $k \geq i + 2$.

Question 5 In the error-reduction theorem for BPP, we start with a probabilistic algorithm that decides a language L with error at most $1/4$ and get a probabilistic algorithm that decides the same language L but with error at most $1/2^n$, where n is the length of the input. We achieve this as follows:

1. Run the algorithm $\mathrm{poly}(n)$ many times; accept only if all executions accept.

2. Run the algorithm $\mathrm{poly}(n)$ many times; accept only if more than half of the executions accept.

3. Run the algorithm $\mathrm{poly}(n)$ many times; stop as soon as it gives the correct answer, which we can detect in polynomial time given the answer.

4. Reducing the error to $1/2^n$ requires $\exp(n)$ many executions.

Question 6 The method of approximate counting via hashing seen in HW4 (and in improved form in the published notes for Week 12) applies to:

1. Any function f defined by $f(x) = |W(x)|$ where $W(x) \subseteq \{0,1\}^{p(|x|)}$ for some polynomial p and the set $\{\langle x, y\rangle : y \in W(x)\}$ is in P.
2. Any function f defined by $f(x) = |W(x)|$ where $W(x) \subseteq \{0,1\}^{p(|x|)}$ for some polynomial p and the set $\{\langle x, y\rangle : y \in W(x)\}$ is in PSPACE.
3. Any function f defined by $f(x) =$ 'the numer of accepting paths of $M(x)$' for a fixed non-deterministic M machine that runs in polynomial space.
4. Any function f defined by $f(x) =$ 'the number of accepting paths of $M(x)$' for fixed non-deterministic M machine that runs in exponential time.

Question 7 In the notes of Week 6 we proved that the Polynomial Identity Testing (PIT) problem for arithmetic expressions has a probabilistic algorithm with bounded error as an application of the Schwartz-Zippel Lemma: Given an arithmetic expression F with variables $x_1, \ldots, x_n$ and constants $+1$ and -1, sample numbers $a_1, \ldots, a_n$ uniformly and independently at random from $\{1, \ldots, N\}$, where $N = \mathrm{poly}(\mathrm{size}(F))$, and check whether $F(a_1, \ldots, a_n) = 0$. Here $\mathrm{size}(F)$ is the number of operations in F.

1. If we chose $N = \log(\mathrm{size}(F))$, 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 $1/4$.
2. If we chose $N = \exp(\mathrm{size}(F))$, then the analysis of the algorithm via the Schwartz-Zippel Lemma would give an error probability as small as inverse exponential in the size of F but it would not be possible to implement the algorithm so that it still runs in time $\mathrm{poly}(\mathrm{size}(F))$.
3. If we chose $N = \exp(\mathrm{size}(F))$, then the analysis of the algorithm via the Schwartz-Zippel Lemma would give an error probability as small as inverse exponential in the size of F, and the algorithm could still be implemented to run in $\mathrm{poly}(\mathrm{size}(F))$.
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 3-colorable:

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

Which one is correct?

1. The problem is in NP since a valid 3-coloring of the graph is a certificate that can be verified in time polynomial in n.
2. The problem is in PSPACE since we can loop through all possible 3-colorings in polynomial space.
3. The problem is in NEXP.
4. None of the above is correct.

Question 9 The Immerman-Szelepzenyi Theorem states that:

1. NTIME(t) = co-NTIME(t)
2. L = NL
3. PSPACE = NPSPACE
4. None of the above.

Question 10 If a language is P-complete then:

1. It is in P and every other problem in P reduces to it by polynomial-time computable reductions.
2. It is in P and every other problem in P reduces to it by logarithmic-space computable reductions.
3. Every problem in P reduces to it by polynomial-time computable reductions but it cannot be in P unless P = NP.
4. Every problem in P reduces to it by logarithmic-space computable reductions but it cannot be in P unless P = NP.

Question 11 Representing TRUE by 1 and FALSE by 0, every Boolean formula $F(x_1, \ldots, x_n)$ with variables $x_1, \ldots, x_n$ may be interpreted as computing a polynomial $P_F(x_1, \ldots, x_n)$ defined according to the following recursion:

$$\begin{array}{lcl} F = x_i & \mapsto & P_F = x_i, \\ F = \text{NOT}(G) & \mapsto & P_F = 1 - P_G, \\ F = \text{AND}(G, H) & \mapsto & P_F = P_G \cdot P_H, \\ F = \text{OR}(G, H) & \mapsto & P_F = 1 - (1 - P_G) \cdot (1 - P_H). \end{array}$$

Exactly one of the statements below is correct. Which one?

1. It is possible to check if a given Boolean formula F is a tautology by testing the polynomial identity $P_F - 1 = 0$ in randomized polynomial time with error probability bounded by $1/4$, by plugging random values $a_1, \ldots, a_n \in \{0, 1\}$ for the variables $x_1, \ldots, x_n$.
2. It is possible to check if a given Boolean formula F is a tautology by testing the polynomial identity $P_F - 1 = 0$ in randomized polynomial time with error probability bounded by $1/4$, by plugging random values $a_1, \ldots, a_n \in \{0, \ldots, 20 \cdot |F|^2\}$ for the variables $x_1, \ldots, x_n$.
3. The first two statements are wrong: the polynomial constructed this way is exponentially big.
4. The first two statements are wrong: the formula F could be a tautology yet $P_F(r_1, \ldots, r_n) \neq 1$ for some real numbers $r_1, \ldots, r_n$.

Question 12 Using the mapping from formulas $F(x_1, \ldots, x_n)$ to polynomials $P_F(x_1, \ldots, x_n)$ given in the previous question, the first step in the sum-check IP protocol to test if F has exactly v satisfying assignments goes as follows: it receives the coefficients $c_0, \ldots, c_d$ of a polynomial $\hat{P}(x_1) = c_0 + c_1 \cdot x_1 + \cdots + c_d \cdot x_1^d$ claimed to agree with the polynomial

$$P(x_1) := \sum_{a_2, \ldots, a_n \in \{0,1\}} P_F(x_1, a_2, \ldots, a_n)$$

and, with arithmetic over the appropriate finite field,

1. checks that $\hat{P}(1) = v + \hat{P}(0)$.

2. checks that $\hat{P}(v+1) = \hat{P}(v) + \hat{P}(1)$.
3. checks that $\hat{P}(0) + \hat{P}(1) = v$.
4. checks that $\sum_{i=1}^{d} \hat{P}(i) = v$.
