CSE491/596, Fall 2021 Problem Set 5 Due Mon. Nov. 22

Reminder: The **Second Prelim Exam** will be on **Wednesday, Dec. 1**, *in class period*. It will have the same rules and in-person logistics (with registered exception) as the first exam.

Lectures and Reading:

Wednesday's lecture will finish proving the PSPACE-completeness of the TQBF problem from ALR chapter 28, section 5. We have already shown that GAP is complete for NL under $\leq_m^{\log}$, and it will be quick to observe that the Circuit Value Problem (CVP) is complete for P under $\leq_m^{\log}$. The remaining pieces covered Friday and next Monday will be: oracle Turing machines and polynomial-time Turing reductions (ALR ch. 28 calls them "Cook reductions"), classical probabilistic computation, and the class BPP. All of these topics are in the last sections 17–19 of Debray's notes, though not in that order. Classical probability will be the springboard for the last five lectures on quantum computation—reading will be given from selected chapters of my textbook with Richard Lipton for over Thanksgiving.

This homework is due on a **Monday**. There will be extended 10–11pm online evening office hours both Wednesday and next Monday before it is due.

———- *Assignment 5, due Nov. 22 "midnight stretchy" on CSE Autograder* ———-

(1) (24 pts.)

Show that the following problem is NP-complete.

Perfect Dominating Set
Instance: An undirected graph G, an integer $k \geq 1$.
Question: Is there a set S of at most k nodes such that every other node is adjacent to *exactly* one node in S?

Note: This is not the same as the standard Dominating Set problem. In a 3-node triangle graph, any one node forms a perfect dominating set, but the 5-node pentagon graph does not have any perfect dominating set of size 2. (There is an ambiguity of whether it is OK for a node in S to have neighbors in S. If yes, then the pentagon graph has perfect dominating sets of size 3. If not, then it has no perfect dominating set at all—the only n-cycle graphs that do are when n is a positive multiple of 3. You may take either interpretation.)

(2) (24 pts.)

Give a polynomial-time many-one reduction from 3SAT to the following mathematical problem:

Instance: A formula for a polynomial p in n variables with integer coefficients.
Question: Does $p(x_1, \ldots, x_n) = 0$ have a solution in which all the x_i are 0 or 1?

This shows that it is NP-hard to solve given systems of equations of the form $p(x_1, \ldots, x_n) = 0$, $x_1^2 = x_1$, $x_2^2 = x_2$, …, $x_n^2 = x_n$ over the real numbers. Show also that this is NP-complete.

Notes: The formula for p need not be "multiplied out" as in $p = x_1^2 - x_2^2 - x_3^2 + 2x_2x_3$—you must also be prepared for the instance p to be given in other forms, such as the "factored

form" $(x_1 - x_2 + x_3)(x_1 + x_2 - x_3)$. The factored forms are what make this problem hard. For your reduction, think of polynomials of the form $q - 1$ where q is in "factored form." You do *not* need to *use* the Cook-Levin Theorem or its proof for this problem, though you may find that some of its properties simplify the reduction a bit. (Don't forget to show why the language of the problem belongs to NP; this is 6 of the 24 points.)

(3) ($5 \times 6 = 30$ pts.) For each of the following stated relationships between complexity classes, say whether it is known to be true or not. In all cases where it is "known," you can prove it by applying theorems that were covered. Where you say "not known," explain why the theorem comparing the two complexity measures involved fails to yield the stated relationship. Note that $\mathsf{E} = \mathsf{DTIME}[2^{O(n)}]$ and $\mathsf{EXP} = \mathsf{DTIME}[2^{n^{O(1)}}]$.

(a) $\mathsf{NSPACE}[(\log n)^2] \subseteq \mathsf{P}$.

(b) $\mathsf{PSPACE} \subseteq \mathsf{E}$.

(c) $\mathsf{NP} \subseteq \mathsf{EXP}$.

(d) $\mathsf{P} \subseteq \mathsf{DSPACE}[n^2]$.

(e) $\mathsf{NSPACE}[O(n)] \subseteq \mathsf{E}$.

(4) (30 pts.)

Consider the following decision problem:

CYCLING DFA
INSTANCE: A DFA $M = (Q, \Sigma, \delta, s, F)$ and a string $x \in \Sigma^n$ where $Q = \{1, 2, \ldots, n\}$.
QUESTION: Does M on input x visit every one of its states and end up back at its start state s?

(a) Sketch a deterministic Turing machine M that decides this problem in $O(\log n)$ space. It is enough to diagram the worktapes of M and say what information each one maintains while sketching the algorithm in pseudocode. (18 pts.)

(b) Now sketch a faster algorithm that uses linear space. You may use the fact that Turing machines can execute mergesort at full $O(N \log N)$-time efficiency in $O(N)$ space, where we may suppose that N is the total bit-length of the given list of binary numbers. Estimate its running time and compare with your answer to (a). (12 pts., for 30 on the problem and 108 total on the set).