CSE596, Fall 2022 Problem Set 4 (short) Due Thu. Oct. 27

Lectures: *For the next four weeks*, lectures will be:

- **Tuesdays 8–9pm online** via Zoom.
- **Thursdays 4–5pm in 201A Capen**. (This is in-person.)
- **Fridays 4–5pm** as normal in O'Brien 112.

The Monday before Thanksgiving (Nov. 21) will also be moved online, not sure if in the same time period. The second prelim exam will probably be after Thanksgiving, on Wed. Nov. 30.

Reading: As of now, the Allender-Loui-Regan (ALR) notes ("Chapters 27 and 28" on the course webpage) on computational complexity become the main text until quantum computing after Thanksgiving. Their order will also be inverted: chapter 28, sections 1–4 on NP-completeness will come before the theorems on complexity class relationships in sections 2.4–2.6 of chapter 27. This is done to make the coverage of polynomial-time mapping reductions follow straight on from that of general mapping reductions. In detail, here is what to read for next week:

1. ALR chapter 27, section 1 and subsections 2.1–2.3. Mostly already covered; note how the Wed. 10/20 lecture shortcutted the issue of time and space constructible functions and O, Θ-notation. Ignore the "Gap" and "Speed-Up" theorems.
2. ALR chapter 28, sections 1 and 2.
3. Then in the middle of reading ALR chapter 28, section 3—before getting to the proof of Cook's Theorem (now called the Cook-Levin theorem)—go back to ALR chapter 27 and skim section 3.3. You do not have to know the "DLOGTIME-uniformity" condition in detail; just appreciate that Theorem 3.1 in ALR ch. 27 represents the idea that "software can be efficiently burned into hardware."
4. Then study the proof of the Cook-Levin theorem in ALR ch. 28, section 3. This circuit-based proof is much briefer than proofs that work directly from Turing machines.
5. Next Friday's lecture may get into section 4 of ALR ch. 28, which will be the main focus of the following week as well.

Please also do read parallel sections of Debray's notes, as well as the course notes, which are close to aligned by week of tern abd MWF date.

————————*Assgt. 4, due Thu. 10/27 "midnight stretchy" on CSE Autograder*————————-

Relative to last year, this could be labeled "Assignment 3b," but it makes more sense for future years to label it this way.

(1) Given a language L, define $L^R = \{x^R : x \in L\}$, where x^R means reversing the string x. Note that if every string in L is a palindrome, then $L = L^R$ automatically, but it is possible to have $L = L^R$ without every string in it being a palindrome, e.g., $L = \{01, 10\}$. Prove that the language of the following decision problem is neither c.e. nor co-c.e.

INSTANCE: A deterministic Turing machine M.

QUESTION: Is $L(M) = L(M)^R$?

Hint: This problem has a high "re-usability" quotient—it is OK to almost violate the University policy against re-submitting previously-used materials, provided you explain how and why the reduction(s) apply to this case. (24 pts.)

(2) A *symmetric* $n \times n$ 0-1 matrix A represents an n-vertex *undirected* graph $G_A = (V, E)$ with $A_{ij} = 1 \iff G$ has an edge between vertex i and vertex j. We'll assume that G_A has no self-loops; i.e., $A_{ii} = 0$ for all i. Represent A in row-major order as a binary string of length $N = n^2$. Define L_2 to be the language of A such that V can be broken into $V_1 \cup V_2$ such that all edges in G_A go between V_1 and V_2.

(a) Show that L_2 belongs to P. You don't have to design a Turing machine M to accept L_2—you can write pseudocode using data structures (such as sets) and appeal to the polynomial-time simulation of a RAM/assembly-program by a TM as covered earlier in the course. As a function of the input length N (not n), what is the asymptotic running time of your pseudocode? (This is how running time is typically measured in algorithms courses. 15+3 = 18 pts.)

(b) Now define L_3 to be the language of undirected graphs G whose vertex set V can be broken into $V_1 \cup V_2 \cup V_3$ such that all edges from V_i go to V_j where $j \neq i$. For example, the 5-vertex "bowtie graph" G with $E(G) = \{(1,2),(2,3),(1,3),(3,4),(3,5),(4,5)\}$ belongs to L_3, but the complete graph on 4 vertices does not. Show that L_3 belongs to NP. (Think about why your approach in the case of L_2 fails to solve this problem. But don't spend a lot of time trying to make a polynomial-time algorithm for L_3, because there are good reasons for believing that no such algorithm exists. 12 pts., making 30 on the problem and 54 on the set)