CSE491/596, Fall 2021 Problem Set 4 Due Wed. Nov. 10

The **Second Prelim Exam** will be on **Wednesday, Dec. 1**, *in class period.* That is, it will be after Thanksgiving. There will also be a homework due on *Monday, Nov. 22*, that is, the Monday before Thanksgiving.

Lectures and Reading: Friday's lecture will finish section 4.4 of ALR chapter 28, and hence finish the topic of NP-completeness. Starting on Monday 11/8, we will give full treatment to space complexity and its relation to time complexity before handling the completeness results in section 5 of that chapter in the third week of November. There is a "hole" in both ALR chapter 27 and in Debray's notes, where ALR states Theorem 2.3 (in the short "Basic Relationships" section 2.4) without giving proofs, while Debray sketches the proof tersely at the top of page 56 (near the end of section 17) but does not make a separate theorem statement. The hole is filled by page 1 of the following handout, which is linked from the course webpage as item 6 of required reading (ignore the dates from Fall 2018 in the first line):

`https://cse.buffalo.edu/~regan/cse491596/CSE596inclusions.pdf`

The remaining pages 2–4 of this handout are heavy going but they tie together material that is heavy going in Debray's notes (split between Theorems 13.7–13.8 and Theorem 18.1) and ALR chapter 27, Theorem 2.5 in section 2.6 (where, however, the proof of the hierarchy theorem for deterministic time is not given, and you should skim/skip the NTIME proof). Thus the reading for next week becomes:

- The rest of Debray, sections 13–18, except we are still postponing the discussion of oracles in the first two pages of section 17, and the TQBF proof will be covered the following week when we rejoin ALR chapter 28 in section 5.
- ALR chapter 27, sections 2.4–2.7.

The reading for the week after next—that is, for the week before Thanksgiving, will be ALR chapter 28 in section 5 in-tandem with the rest of Debray's notes.

——— *Assignment 4, due Nov. 10 "midnight stretchy" on CSE Autograder* ———

(1) For any language L, define *Halves*(L) to be the language of prefixes w of strings $x \in L$ s.t. w includes at least the first half of x. (In symbols, $w \sqsubseteq x$ means w is a prefix of x. The other requirement is $|w| \geq |x|/2$ for some $x \in L$.) Also define *Prefs*(L) to be the set of all prefixes of strings in L, i.e. without the "$|w| \geq |x|/2$" restriction.

(a) Prove that if L belongs to P, then *Halves*(L) belongs to NP.

(b) Prove however that when L is a suitable encoding of the valid-computation checking predicate (the "Kleene T predicate") then the language *Prefs*(L) is undecidable, even though L belongs not only to P but even to deterministic linear time.

(c) Also answer: Is *Prefs*(L) always c.e. when L is c.e.? ($6 + 12 + 6 = 24$ pts.)

(2) Now define the following language S, which belongs to the class P:

$$S = \{\langle\phi\rangle @^n a \langle\phi\rangle : \phi \text{ is a 3CNF formula in } n \text{ variables}, a \in \{0,1\}^n, \text{ and } \phi(a) = \mathsf{true}\}.$$

The reason this does not put angle brackets around the whole input is that while we do not care exactly how the formula is encoded, we do care about the encoding detail that the candidate assignment a to the variables as a simple binary string comes before a second copy of the formula. Plus we stuck a string of n @-signs before the assignment—why did we do that? The reason is that if $x = \langle\phi\rangle @^n a \langle\phi\rangle$, then $w = \langle\phi\rangle @^n$ is exactly the first half of x. This sets up the way $Halves(L)$ is defined in problem (1).

The reason why S is in P is that we can quickly check that ϕ has the stated number n of variables and then evaluate ϕ on the *given* assignment a. But what about testing whether a given string of the form $w = \langle\phi\rangle @^n$ belongs to $Halves(S)$? **Show that $Halves(S)$ is NP-complete.**

Also show that if we really had a polynomial-time Turing machine M such that $L(M) = Halves(S)$, then given any formula ϕ, we could use M repeatedly to **find** a satisfying assignment whenever one exists. (The basic idea, often called "reducing search to decision," was also used in the 10/28 lecture example showing how deciding the language "FACT" enables one to find a factor. $9 + 9 = 18$ pts.)

(3) Given a graph $G = (V, E)$ that includes a triangle of edges $(u, v), (u, w), (v, w)$, say that a node t "abuts the triangle" if t borders one of its nodes—that is, if (t, u), (t, v), or (t, w) also is an edge in E. Show *by reduction from 3SAT* that the following decision problem is NP-complete. (24 pts.)

INSTANCE: An undirected graph $G = (V, E)$ and an integer $k \geq 1$.

QUESTION: Is there a subset S of at most k nodes that abuts every triangle in G?

(4) Suppose a university degree has m requirements $R_1, \dots, R_m$, such that each R_j can be satisfied by any one of a set S_j of courses. The sets S_j may overlap—that is, one course might satisfy multiple requirements (like being both a CSE breadth course and a pre-req. for a depth course you want to take). Show *by reduction from 3SAT* that the following problem is NP-complete:

INSTANCE: A list of sets S_j of courses, so that any member of S_j satisfies a corresponding requirement R_j, and a number $k \geq 0$.

QUESTION: Is there a way to take just k courses so that all requirements are satisfied?

Show that this problem (in general, apart from real universities) is NP-complete. (Hint: you can do this when each requirement R_j has just two courses that can satisfy it, so R_j is like an edge in a graph whose nodes are the courses. 24 pts., for 90 total on the set)