# CSE491/596, Fall 2021 Problem Set 2 Due Wed. Oct. 6

The **First Prelim Exam** will be held *in-person, in class period* on **Wed. Oct. 13**. For those who must take the exam remotely, alternate and simultaneous provision will be made. It will cover the domain of the first two assignments and also factual material about decidable and undecidable languages. It will not include proofs by reduction, even though they may be lectured on (just) before the exam.

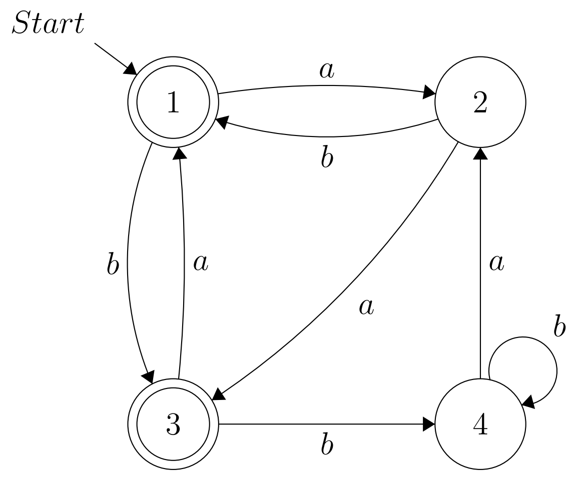
Note that this is a longer assignment. Last year I made a shorter assignment with Myhill-Nerode type problems. The difference in the schedule this year (no Labor Day lecture) and the glitch with Wednesday's lecture on Myhill-Nerode proofs leads me not to do so with a one-week timeframe. But this puts extra responsibility not to leave it all to the last minute.

**Lectures and Reading** (same as said last year). For next week, read section 7 of Debray's notes. Then jump way ahead to section 13 to get the definition of time complexity, but stop at Theorem 13.6 (note that Theorem 13.4 is basically the same as the $\{a^n b^n\}$ example already started in lecture. The definition of space complexity is wedged in section 17 as Definition 17.5, but it's simpler to just read it here: the *space used* by a deterministic Turing machine $M$ on an input $x$ is the number of tape cells that are ever *changed* to a *different* character. Then $M$ runs within space $s(n)$ if for all $n$ and inputs $x$ of length $n$, $M$ on input $x$ uses no more than $s(n)$ space. If $M$ on input $x$ does not halt, it is conventional to regard the space as well as time used to be infinite, but one can also define terms carefully to avoid needing to rely on this convention. *Then go back* and read section 6, which you'll notice already refers to "time" and "space" as if they were known informally from an algorithms course or somesuch. Finally, for next week, read section 8, but pause before the definition of mapping reductions and the examples on page 26, since we will go over to the notes by John Watrous in parallel with reading page 25.

The attitude of lectures will be to de-emphasize the character-level details of Turing machines once we show how to build a universal Turing machine—one that is fairly time-efficient, in fact. That is to say, we focus on a basic vocabulary of TM operations that suffice to emulate a rudimentary assembly language—for which see my hand-drawn handout https://rjlipton.files.wordpress.com/2014/05/utmramsimulator.png, which is enough to establish the principle of how TMs are equivalent to high-level programming languages. Once we have it, we dispense with char-level drawings of Turing machines and allow them to be *described in pseudocode*—provided the pseudocode is low-level enough to indicate the time and space complexity up to linear (or at least polynomial) factors.

———- *Assignment 2, due Oct. 6 "midnight stretchy" on CSE Autograder* ———-

(1) Calculate a regular expression for the language of the following deterministic finite automaton. (The picture was drawn using the Finite State Machine Designer by Evan Wallace.)



It has $Q = \{1,2,3,4\}$, $\Sigma = \{a,b\}$, $s = 1$, $F = \{1,3\}$, and

$$\delta = \{(1,a,2),(1,b,3),(2,a,3),(2,b,1),(3,a,1),(3,b,4),(4,a,2),(4,b,4)\}.$$

You must show the steps of the algorithm used in class. You may re-number the states if you wish. (18 pts.)

(2) Now define $N$ to be the finite automaton obtained by reversing each arc of $M$ (but without changing the start state or final states). Note that $N$ is properly an NFA: it has nondeterminism at states 2 and 4. Convert $N$ into an equavalent DFA $M'$. Then use your DFA to find a string $z$ of length 7 that $N$ can process from 1 only back to state 1. $(18+6 = 24$ pts. Your $M'$ will be larger than 8 states but won't go all the way to 16.)

(3) Use the Myhill-Nerode proof script to show that the following languages over $\Sigma = \{0,1\}$ are not regular. The default assumption on natural numbers $m$ and $n$ is that they can be zero, i.e., $m, n \geq 0$. Recall that $\#c(x)$ means the number of occurrences of the character $c$ in the string $x$.

(a) $L_a = \{(00)^m 10^n : n = 2m\}$.

(b) $L_b = \{(00)^m 10^n : n > m\}$.

(c) $L_c = \{v10w : \#0(v) = \#1(w)\}$.

The points total 30 for all three languages together.

(4) Prove that for any $k \geq 0$, $\{0,1\}^k$ is a PD set of size $2^k$ for the language of palindromes over $\Sigma = \{0,1\}$. (12 pts.)

(5) Design on paper a Turing machine $M$ such that $L(M) = L_a$ in problem (3). Here it may help you to simplify the definition. You may design either a 1-tape or a 2-tape Turing machine, but for the final 1 pt., should state whether it runs in linear time or in more-than-linear time. (16 pts., for 100 total on the set)