Extra Practice Problems 9

This handout contains a bunch of problems that we hope will serve as a good cumulative review for all the material that we've covered this quarter. If there are any other topics you'd like some additional practice with, please let us know!

The latter problems in this set of practice problems touch on topics we haven't covered yet as of the time this set of problems is released (verifiers, non-**RE** languages, **P** and **NP**), so don't panic if you haven't seen those concepts yet.

We'll release solutions on Wednesday.

Problem One: Set Theory

Prove or disprove: if A, B, C, and D are sets where $A \times B \subseteq C \times D$, then $A \subseteq C$ and $B \subseteq D$.

Problem Two: Induction

In many applications in computer science, especially cryptography, it is important to compute exponents efficiently. For example, the RSA public-key encryption system, widely used in secure communication, relies on computing huge powers of large numbers. Fortunately, there is a fast algorithm called *repeated squaring* for computing x^y in the special case where y is a natural number.

The repeated squaring algorithm is based on the following function *RS*:

$$RS(x, y) = \begin{cases} 1 & \text{if } y = 0 \\ RS(x, y/2)^2 & \text{if } y \text{ is even and } y > 0 \\ x \cdot RS(x, (y-1)/2)^2 & \text{if } y \text{ is odd and } y > 0 \end{cases}$$

For example, we could compute 2^{10} using *RS*(2, 10) as follows:

In order to compute *RS*(2, 10), we need to compute *RS*(2, 5)^2.
In order to compute *RS*(2, 5), we need to compute 2·*RS*(2, 2)^2.
In order to compute *RS*(2, 2), we need to compute *RS*(2, 1)^2.
In order to compute *RS*(2, 1), we need to compute 2·*RS*(2, 0)^2.
By definition, *RS*(2, 0) = 1
so *RS*(2, 1) = 2·*RS*(2, 0)^2 = 2·1^2 = 2.
so *RS*(2, 2) = *RS*(2, 1)^2 = 2^2 = 4.
so *RS*(2, 5) = 2·*RS*(2, 2)^2 = 2·4^2 = 32.
so *RS*(2, 10) = *RS*(2, 5)^2 = 32^2 = 1024.

The *RS* function is interesting because it can be computed much faster than simply multiplying x by itself y times. Since *RS* is defined recursively in terms of *RS* with the y term roughly cut in half, *RS* can be evaluated using approximately $\log_2 y$ multiplications. (You don't need to prove this).

Prove that for any $x \in \mathbb{R}$ and any $y \in \mathbb{N}$, that $RS(x, y) = x^y$. *(Hint: use complete induction on y.)*

Problem Three: Graphs

Recall from the second midterm exam that if $G = (V, E)$ is an undirected graph, then a ***dominating set in G*** is a set D where every node $v \in V$ either belongs to D or is adjacent to a node in D (or both).

Now, let's introduce some new terminology. A ***domatic partition of G*** is a way of splitting the nodes in G into disjoint, nonempty sets V_1, V_2, …, V_n such that each set V_i is a dominating set. (Two sets S and T are disjoint if $S \cap T = \varnothing$.) The ***domatic number*** of G, denoted $d(G)$, is the maximum number of sets in any domatic partition of G.

i. Let G be an undirected graph and let δ be the minimum degree of any node in G. Prove that $d(G) \leq \delta + 1$.

An ***isolated node*** in a graph G is a node that is not adjacent to any other nodes in G.

ii. Let G be an undirected graph with no isolated nodes. Prove that $d(G) \geq 2$. *(Hint: Use a result from the practice second midterm exam.)*

iii. Prove that the bounds you came up with in parts (i) and (ii) are "tight" in the sense that, in general, you cannot improve upon these upper bounds or lower bounds without more knowledge of the structure of the graph. Specifically, give a graph G where $d(G) = \delta + 1$ and give a graph G with no isolated nodes where $d(G) = 2$. Briefly justify your answers.

Problem Four: First-Order Logic

Given the predicates

- *String*(w), which states that w is a string over alphabet Σ;
- *TM*(M), which states that M is a TM with input alphabet Σ; and
- *Accepts*(M, w), which states that M accepts w,

along with the function $\langle O \rangle$, which represents the encoding of some object O, write a statement in first-order logic that says "$L_D \notin$ **RE**." (We'll cover L_D on Wednesday, May 31. Looking forward: the language L_D is defined as $L_D = \{ \langle M \rangle \mid \langle M \rangle \notin \mathscr{L}(M) \}$)

Problem Five: Binary Relations

Let $A = \{1, 2, 3\}$. Draw the Hasse diagram of the lift of $<$ over A to $\wp(A)$, which is the relation $\tilde{<}$ defined as

$$X \tilde{<} Y \quad \text{if} \quad Y \neq \varnothing \text{ and for any } x \in X \text{ and } y \in Y, \text{ we have } x < y.$$

Problem Six: Functions and Cardinality

Let A, B, C, and D be sets where $|A| = |C|$, $|B| = |D|$, $A \cap B = \varnothing$, and $C \cap D = \varnothing$. Using the formal definition of equal cardinality, prove that $|A \cup B| = |C \cup D|$.

Problem Seven: The Pigeonhole Principle

Suppose that you have a set S of $n > 0$ natural numbers. Prove that there must be a nonempty subset of S where the sum of the numbers in that subset is a multiple of n. *(Hint: Number the elements of S as $x_1, x_2, \ldots, x_n$. Then, look at x_1, $x_1 + x_2$, $x_1 + x_2 + x_3$, etc.)*

Problem Eight: DFAs and NFAs

Here's some true-or-false questions to ponder:

i. True or false: If D is a DFA over alphabet Σ and D has no accepting states, then $\mathscr{L}(D) = \varnothing$.

ii. True or false: If D is a DFA over alphabet Σ and D has no rejecting states, then $\mathscr{L}(D) = \Sigma^*$.

iii. True or false: If N is an NFA over alphabet Σ and N has no accepting states, then $\mathscr{L}(N) = \varnothing$.

iv. True or false: If N is an NFA over alphabet Σ and N has no rejecting states, then $\mathscr{L}(N) = \Sigma^*$.

Let $\Sigma = \{\texttt{a}, \texttt{b}, \texttt{c}, \texttt{d}, \texttt{e}\}$ and let L be the following language:

$$L = \{ w \in \Sigma^* \mid \text{every character from } \Sigma \text{ appears at least once in } w \}$$

Any DFA for L must have at least 32 states (you don't need to prove this.)

v. Prove that any DFA for $\overline{L}$ must have at least 32 states.

vi. Design a reasonably-sized NFA for $\overline{L}$. This shows that even if you can't find a small NFA for a language, you might be able to find a small NFA for its complement.

Problem Nine: Nonregular Languages

Let $\Sigma = \{\texttt{a}, \texttt{b}\}$ and consider the language $L = \{ wx \mid w \in \Sigma^*, x \in \Sigma^*, |w| = |x|, \text{ and } w \neq x \}$. Prove that L is not a regular language.

Problem Ten: Context-Free Grammars

This question explores closure properties of CFLs.

i. Show that the context-free languages are closed under union, concatenation, and Kleene star.

ii. Although we didn't prove this, the context-free languages are not closed under complementation. In lecture, you saw a CFG for the language $\{ w \in \{\texttt{a}, \texttt{b}\}^* \mid w \text{ is a palindrome} \}$, and on Problem Set Seven you built a CFG for the complement of this language. Explain how this is possible even though the context-free languages aren't closed under complementation.

Problem Eleven: Turing Machines

Design a TM over the alphabet $\Sigma = \{\texttt{a}, \texttt{b}\}$ whose language is $\{ w \in \Sigma^* \mid w$ does not contain `aa` or `bb` as substrings $\}$.

Problem Twelve: R and RE Languages

(We will cover the material necessary to solve this problem on Wednesday, May 31st.)

Prove that there is a language X where $X \subseteq L_D$, where X contains infinitely many strings, and where X is an **RE** language.

Problem Thirteen: Impossible Problems

Let $L = \{ \langle M \rangle \mid M \text{ is a TM and } \mathscr{L}(M) = \{ \langle M \rangle \} \}$. In other words, L is the set of all TMs that accept themselves and only themselves. (We can think of them as narcissistic TMs.)

Prove that $L \notin \mathbf{R}$.