Extra Practice Problems 7

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!

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:

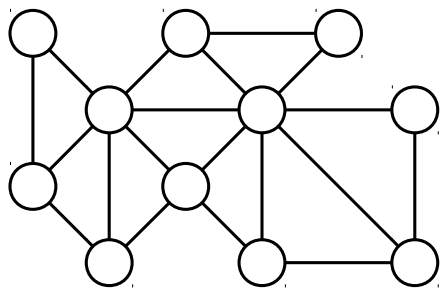
In order to compute $RS(2, 10)$, we need to compute $RS(2, 5)^2$.
In order to compute $RS(2, 5)$, we need to compute $2 \cdot RS(2, 2)^2$.
In order to compute $RS(2, 2)$, we need to compute $RS(2, 1)^2$.
In order to compute $RS(2, 1)$, we need to compute $2 \cdot RS(2, 0)^2$.
By definition, $RS(2, 0) = 1$
so $RS(2, 1) = 2 \cdot RS(2, 0)^2 = 2 \cdot 1^2 = 2$.
so $RS(2, 2) = RS(2, 1)^2 = 2^2 = 4$.
so $RS(2, 5) = 2 \cdot RS(2, 2)^2 = 2 \cdot 4^2 = 32$.
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

Consider the following graph:



i. Is this graph 4-colorable? Justify your answer.

ii. Is this graph 3-colorable? Justify your answer.

iii. Is this graph 2-colorable? Justify your answer.

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**."

Problem Five: Functions

Prove that $|A_{TM}| = |\Sigma^*|$. *(Hint: Use the Cantor-Bernstein-Schröeder theorem and consider a TM that accepts all strings.)*

Problem Six: Binary Relations

A ***strict partial order*** is a binary relation R over a set A with the following properties:

- R is ***asymmetric:*** $\forall a \in A.\ \forall b \in A.\ (aRb \to \neg(bRa))$
- R is transitive.

This question explores strict partial orders.

i. Prove that if R is strict partial order over a set $A \neq \emptyset$, then R isn't a partial order over A.

ii. Let R be a strict partial order over A and define a new binary relation T over A as follows:

$$aTb \text{ if } aRb \text{ or } a = b$$

Prove that T is a partial order.

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 $\mathscr{L}(D) = \emptyset$.

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 $\mathscr{L}(N) = \emptyset$.

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

Let $\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}, \mathtt{e}\}$ 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. Though it's much harder to prove this, every NFA for L must also have at least 32 states. Explain, intuitively, why nondeterminism doesn't help reduce the number of states here.

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

vii. 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 $\Sigma = \{\mathtt{a}, \mathtt{b}\}$ 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 $\{ w \in \{\mathtt{a}, \mathtt{b}\}^* \mid w \text{ is a palindrome} \}$, and on Problem Set Six 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 $\Sigma = \{\mathtt{a}, \mathtt{b}\}$ whose language is $\{ w \in \Sigma^* \mid w$ does not contain `aa` or `bb` as substrings $\}$.

Problem Twelve: R and RE Languages

Many languages that aren't **RE** languages have the interesting property that even though the language itself isn't **RE**, there is a subset of the language containing infinitely many strings that is an **RE** language. Such languages are called ***nonimmune languages***.

i. Prove that EQ_{TM} is nonimmune.

ii. Prove that L_{D} is nonimmune.

Problem Thirteen: Impossible Problems

Let $L = \{ \langle M \rangle \mid M$ is a TM and $\mathscr{L}(M) = \{ \langle M \rangle \} \}$. Prove that $L \notin$ **RE**.

Problem Fourteen: P and NP

Suppose that the following claim is true:

If L_1 and L_2 are **NP** languages other than $\emptyset$ or Σ^*, then $L_1 \leq_{\text{p}} L_2$

Decide which of the following statements is true and briefly justify your answer:

- In this case, **P** is definitely equal to **NP**.
- In this case, **P** is definitely not equal to **NP**.
- In this case, **P** may or may not be equal to **NP**.