CS103 Handout 36
Fall 2015 November 23, 2015

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!

We'll release solutions once we get back from the break.

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*:

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

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

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 $V_1, V_2, \ldots, V_n$ such that each set V_i is a dominating set. (Two sets S and T are disjoint if $S \cap T = \emptyset$.) 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 $d(G) \le \delta + 1$.

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 $d(G) \ge 2$. *(Hint: What did you prove on the first 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**."

Problem Five: Binary Relations

Let A be an arbitrary set and $<_A$ be an arbitrary strict order over A. We'll say that a ***chain*** in A is a series of elements $x_1, x_2, \ldots, x_k$ in A such that

$$x_1 <_A x_2 <_A \ldots <_A x_k.$$

Intuitively, a chain is a series of values in ascending order according to the strict order $<_A$. The ***length*** of an chain is the number of elements in that chain.

i. Consider the $\subset$ relation over the set $\wp(\{a, b, c\})$. What is the length of the longest chain in this strict order? Give an example of a chain with that length. No justification is necessary.

An ***antichain*** is a set $X \subseteq A$ such that no two elements in X can be compared by the $<_A$ relation. In other words, a set $X \subseteq A$ is an antichain if for any $a, b \in X$, both $a <_A b$ and $b <_A a$ are false. The ***size*** of an antichain X is the number of elements in X.

ii. Consider the $\subset$ relation over the set $\wp(\{a, b, c\})$. What is the size of the largest antichain in this strict order? Give an example of an antichain with that size. No justification is necessary.

Given an arbitrary strictly ordered set, you can't say anything a priori about the size of the largest chain or antichain in that strict order. However, you can say that at least one of them must be relatively large relative to the strictly ordered set. Let r and s be natural numbers. We're going to ask you to prove the following result: if $|A| = rs+1$, then either A contains a chain of length $r+1$ or an antichain of size $s+1$. The propositional equivalence $P \vee Q \equiv \neg P \to Q$ will be useful here. To prove at least one of P or Q is true, you can instead prove that if P is false, then Q is true. For the purposes of this problem, we're going to prove that if A does ***not*** contain an chain of length at least $r+1$, it does contain an antichain of size at least $s+1$.

iii. For each element $a \in A$, we'll say that the ***height*** of a is the length of the longest chain whose final element is a. Prove that if A does not contain a chain of length $r+1$ or greater, then there must be at least $s+1$ elements of A at the same height.

iv. Your result from part (iii) establishes that there must be a collection of $s+1$ elements of A at the same height as one another. Let X be any set of $s+1$ such elements. Prove that X must be an antichain.

Intuitively speaking, if $<_A$ is a strict order over A that represents some prerequisite structure on a group of tasks, a chain represents a series of tasks that have to be performed one after the other, and an antichain represents a group of tasks that can all be performed in parallel (do you see why?) In the context of parallel computing, the result you've proved states that if a group of tasks doesn't contain long dependency chains, that group must have a good degree of parallelism.

Problem Six: Functions

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

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. 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 $\Sigma = \{\mathtt{a}, \mathtt{b}\}$ 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 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 $\Sigma = \{\mathtt{a}, \mathtt{b}\}$ whose language is $\{\ w \in \Sigma^* \mid w \text{ does not contain } \mathtt{aa} \text{ or } \mathtt{bb}$ as substrings $\}$.

Problem Twelve: R and RE Languages

On Problem Set Seven, you saw that some nonregular languages have infinite regular languages as subsets, while others do not. This problem generalizes this result to undecidable and unrecognizable languages.

i. Prove that there is an infinite decidable subset of EQ_{TM} (defined on Problem Set Nine).

ii. Prove that there is an infinite recognizable subset of L_{D}.

Problem Thirteen: Impossible Problems

Let $L = \{ \langle M \rangle \mid M$ 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 $L \notin$ **RE**.

Problem Fourteen: P and NP

Suppose that the following claim is true:

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

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**.