Computer Science 345
The Efficient Universe

Final Exam

No collaboration is permitted on the final. You may refer to your own notes, textbooks, and online references. You are not supposed, however, to search for solutions online.

Your solutions are due by 4:00pm on Friday, May 19. You may work on the exam for up to 24 consecutive hours. Since students will be taking the exam at different times, you should not discuss the final exam with others till that date. Please, return the exam to Donna O'Leary (410 CS Building), or email a pdf or word file to kmakaryc AT cs.princeton.edu.

Make your answers as clear, precise and brief as you can. No problem requires more than a page in clear handwriting (we recommend to write a draft first). You can give partial answers (or solve special cases) for possible partial credit. If you make any assumptions state them clearly. You can use theorems/lemmas/facts from the class, precept, textbook and recommended references. You should prove any other theorems/lemmas/facts you use.

You may ask questions about the course topics up until the time you start working on the exam. After that, you should only ask questions if you need a clarification about a problem on the final. We will try to answer all your questions.

Each of the problems below is worth 25 points. Your grade will be the total of the grades on your best 5 of the 6 problems.

Hint: *Try to solve easy problems first and then proceed to more difficult ones.*

Problem 1 For each of the following statements, prove that it is true or prove that its negation is true.

1. If L is a decidable set of pairs (x, y) of strings, then the language

   $$M = \{x : \forall y \ \ (x, y) \in L\}$$

   is also decidable.

2. If $\mathcal{P} = \mathcal{NP}$, then $\mathcal{BPP} = \mathcal{P}$.

3. If $\mathcal{NP} = \mathcal{BPP}$, then $\mathcal{NP} = co\mathcal{NP}$.

4. Every language belongs to $\mathcal{NP} \cup co\mathcal{NP}$.

5. Let L be a decidable set. Define its characteristic sequence $x_1, x_2, \ldots$ as follows:

   $$x_i = \begin{cases} 0 & , \text{ if } i \notin L \\ 1 & , \text{ if } i \in L. \end{cases}$$

   where i is viewed as a binary string.

   Prove that, there exists a constant C (depending on L) such that for every natural n

   $$K(x_1 x_2 \ldots x_n) \leq \log n + C.$$

   Here $K(x_1 x_2 \ldots x_n)$ denotes the Kolmogorov complexity of the binary string $x_1 \ldots x_n$.

Problem 2 Let p be a prime, and $+$, $-$ and $\times$ denote the addition, subtraction and multiplication operations modulo p. You are given five subroutines (black boxes), Plus, Minus, Times, Random and IsZero. The first three subroutines compute sum, difference and product of two numbers respectively. The fourth function, Random, generates a random uniform number in $\mathbb{Z}_p$. Finally, IsZero compares a number with zero and returns true or false.

However, the subroutine Times may err, but we are guaranteed that it happens on at most 1 percent of all inputs. In other words, for (uniform) random, independent $y \in_R \mathbb{Z}_p$ and $z \in_R \mathbb{Z}_p$,

$$\Pr\left(\text{Times}(y, z) \neq y \times z\right) \leq \frac{1}{100}.$$

The other four subroutines always work correctly.

Design a probabilistic algorithm T that for *every* possible input pair (y, z) computes $y \times z$ with error probability at most 1 percent:

$$\Pr\left(T(y, z) \neq y \times z\right) \leq \frac{1}{100},$$

where the probability is over the internal coin tosses of T. The algorithm can call the five subroutines described above; each call is an atomic operation and takes one unit of time. The algorithm should run in constant time (*i.e.* it can make a constant number of calls to the subroutines and perform a constant number of other operations).

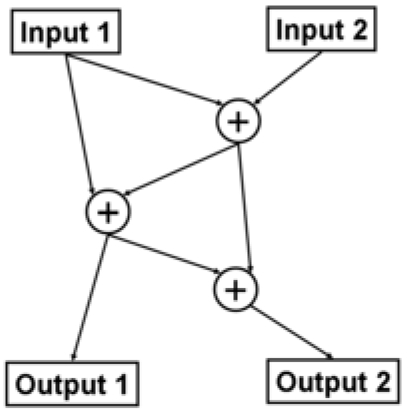
Hint: It is better to think of p as unknown.

Hint: Start by finding an algorithm whose error is at most 4/100 on any input pair.

Problem 3 Recall that Circuit Satisfiability (the set of all satisfiable circuits) is an $\mathcal{NP}$-complete language. The task of this problem is proving that a variant of this problem, called Planar Circuit Satisfiability is also $\mathcal{NP}$-complete.

A graph is *planar* if it can be drawn in the plane with no edges crossing. A circuit is planar if its underlying graph is planar. The Planar Circuit Satisfiability Language is the set of all satisfiable *planar* circuits with *AND*, *OR*, *NOT* and *XOR* gates (notice the additional *XOR* gate).

- Determine the function computed by the following "gadget" circuit with two inputs "Input 1", "Input 2" and two outputs "Output 1", "Output 2".



The symbol $\oplus$ denotes the XOR gate.

Namely if "Input 1" is a and "Input 2" is b, what are the outputs?

- Use the gadget above to give a reduction from Circuit Satisfiability to Planar Circuit Satisfiability.

- Conclude the proof that Planar Circuit Satisfiability is $\mathcal{NP}$-complete.

Problem 4

Assume that $G : \{0,1\}^k \rightarrow \{0,1\}^n$ is an efficient (s, ε) pseudo-random generator (*i.e.*, no circuit of size s can distinguish a random n-bit string from G's output on a random k-bit string with probability more than ε).

- State precisely how Alice and Bob, who share a random k-bit string K, may use it as a "computational" one-time pad, for Alice to secretly transmit an n-bit message M to Bob. In what sense will an eavesdropper Eve (who has no information about K or M, but has access to all transmitted bits) will not know M? Assume that Eve's actions are computable by a circuit of size s.
- Suggest a way for iterating G twice (or several times) so as to increase its output length. Explain why you expect it to still be a pseudo-random generator (for some reasonable parameters depending on s and ε), and what method of proof can be used to show that.
- Conclude by explaining how the key K can be used to transmit several n bit messages securely, and not just one.

Remark: In all parts, you can refer to the previous parts even if you do not solve them.

Problem 5 Assume that $\mathcal{NP} \cap co\mathcal{NP} = \mathcal{P}$. Prove that the Discrete Logarithm problem has a polynomial time algorithm. Note that the assumption refers to languages (decision problems), and the conclusion refers to a function with non-boolean output.

Explain the impact of the assumption on cryptography.

Problem 6

- Give an example of an undecidable language, which can be computed by a family of polynomial size circuits.
- Prove the existence of a decidable language, which cannot be computed by a family of polynomial size circuits.