

Problem 1

In this problem you prove an analog of the Karp-Lipton-Sipser theorem for counting problems.

We say that a circuit C solves #3SAT on n variables if for every 3CNF ϕ on n variables, $C(\phi)$ outputs the number of satisfying assignments for ϕ . We say #3SAT has a polynomial-size family of circuits if there is a polynomial p such that for every n , there exists a circuit of size $p(n)$ that solves #3SAT on n variables.

- (a) Show that the following decision problem is in coNP:

$$L = \{(C, 1^n) : C \text{ solves #3SAT on } n \text{ variables}\}$$

- (b) Show that if #3SAT has a polynomial-size family of circuits, then $P^{\#3SAT} = \Sigma_2$.

Problem 2

Let $\mathbb{F}_2 = \{0, 1\}$ be the two-element field. A function $f : \mathbb{F}_2^n \rightarrow \mathbb{F}_2$ is *linear* if it satisfies the condition $f(x + y) = f(x) + f(y)$ for all $x, y \in \mathbb{F}_2^n$. Consider the following *linearity test* for f :

Choose random $x, y \sim \mathbb{F}_2^n$ and accept iff $f(x) + f(y) = f(x + y)$.

When f is linear, clearly the test accepts with probability 1. In this problem you will show that if f is δ -far from linear, then the test rejects with probability at least $\delta/3$.

You will prove the contrapositive. Suppose that the test accepts f with probability $> 1 - \delta/3$. Let $\ell : \mathbb{F}_2^n \rightarrow \mathbb{F}_2$ be the function

$$\ell(x) = \begin{cases} 1, & \text{if } \Pr_y[f(x + y) + f(y) = 1] > 1/2 \\ 0, & \text{if } \Pr_y[f(x + y) + f(y) = 0] > 1/2. \end{cases}$$

- (a) Show that for at least a $1 - \delta$ fraction of $x \in \mathbb{F}_2^n$, $\Pr_y[f(x) = f(x + y) + f(y)] > 2/3$, so in particular f is δ -close to ℓ .
- (b) Show that ℓ is a linear function.

Hint: For every $x, y \in \mathbb{F}_2^n$, show that there exists some $z \in \mathbb{F}_2^n$ such that $\ell(x) = f(z) + f(x + z)$, $\ell(y) = f(z) + f(y + z)$, and $\ell(x + y) = f(x + z) + f(y + z)$.

Instance checkers Suppose you have an algorithm that solves some difficult problem, but you are not sure if the algorithm is correct. Ideally, you would want to first check the correctness of the algorithm, then use it to solve the instance at hand. An instance checker is a procedure that combines both these steps into one.

An *instance checker* for a problem f (this could be a decision problem, or a counting problem) is a randomized polynomial-time algorithm I that has oracle access to a candidate algorithm A for f and has the following behavior:

- For every A and every x , $I^A(x)$ outputs either an answer or the special symbol "fail".
- If $A(x) = f(x)$ for every x , then $I^A(x) = f(x)$ for every x . (If A is a good algorithm for f , then I^A always outputs the correct answer.)
- For every A and every x , $\Pr[I^A(x) \in \{f(x), \text{"fail"}\}] \geq 3/4$. (If A is not a good algorithm for f , then with high probability I^A either fails or outputs the correct answer.)

Problem 3

In \mathbb{F}_2 -matrix multiplication you are given two $n \times n$ matrices A, B over \mathbb{F}_2 and want to compute the matrix product AB . The best known algorithm for matrix multiplication takes time $\Omega(n^{2.3})$. Show that matrix multiplication has an instance checker that runs in time $\tilde{O}(n^2)$. (Assume you have random access to the input.)

Hint: Check the answer by multiplying it with a random vector.

Problem 4

In this problem you investigate instance checkers for #P-complete problems.

- (a) Show that every #P-complete problem has an instance checker.

Hint: Look at the interactive proof for #3SAT. The prover in this protocol can be realized by a polynomial-time algorithm with oracle access to #3SAT. Let I play the role of the verifier and A play the role of the prover.

- (b) Let f be a #P-complete problem. Show that there is a randomized algorithm A for f with the following property. For every (not necessarily efficient) randomized algorithm M that solves f and every input x , if $M(x)$ halts within t steps, then $A(x)$ halts within $p_M(|x|, t)$ steps, where p_M is some polynomial whose coefficients may depend on the description of M but not on x or t .

(We can think of A as an "almost optimal" randomized algorithm for f , since its running time of A is not much worse than that of the "best" randomized algorithm for f .)

Hint: Your algorithm A should run the instance checker for f , using candidate algorithms M as the oracle.