COMS 4236 – Test 3 Practice Problems

(1) Show that for every $L \subseteq \{0,1\}^*$, if $L \in \mathsf{RP}$, then $L \in \mathsf{P/poly}$ (that is, L has polysize circuits) in three steps.

(a) Since $L \in \mathsf{RP}$, there is a deterministic polynomial-time machine $M(x,r)$ and a polynomial p such that for every $x \in \{0,1\}^n$,

$$x \in L \implies \Pr_{r \in \{0,1\}^{p(n)}}[\, M(x,r) = 1 \,] \geq \tfrac{1}{2},$$
$$x \notin L \implies \Pr_{r \in \{0,1\}^{p(n)}}[\, M(x,r) = 1 \,] = 0.$$

Choose t strings $r_1, \ldots, r_t \in \{0,1\}^{p(n)}$ independently and uniformly at random, and accept x if and only if $M(x, r_i) = 1$ for at least one i. Show that an input $x \notin L$ is never accepted, and that an input $x \in L$ is rejected with probability at most 2^{-t}.

(b) Set $t = n+1$. Show that there exist *fixed* strings $r_1, \ldots, r_t$ such that for every $x \in \{0,1\}^n$,

$$x \in L \iff M(x, r_i) = 1 \text{ for some } i \in \{1, \ldots, t\}.$$

In fact, show that at least half of all choices of $(r_1, \ldots, r_t)$ have this property.

Hint: Which inputs can a choice of $(r_1, \ldots, r_t)$ get wrong? How many such inputs are there at most?

(c) Now using (b), show that $L \in \mathsf{P/poly}$. That is, describe a polysized circuit family C_n, $n > 0$ that decides L on inputs of length n. What is hardwired into it, and what is its size in terms of the running time of M?

(d) Now show how you can adapt this argument to prove the stronger statement that for every $L \in \mathsf{BPP}$, $L \in \mathsf{P/poly}$.

Hint: Two things change: the rule for combining the t runs, and the bound on the probability that the combined answer is wrong. You may use the following Chernoff bound: if $X_1, \ldots, X_t$ are independent 0/1 random variables, each with mean at least 2/3, then $\Pr[\sum_i X_i \leq t/2] \leq e^{-t/18}$.

Solution. (a) If $x \notin L$ then $M(x,r) = 0$ for every r, so no choice of $r_1, \ldots, r_t$ accepts x. If $x \in L$, each r_i independently has $M(x, r_i) = 0$ with probability at most $1/2$, so all t runs reject with probability at most 2^{-t}.

(b) Say that $(r_1, \ldots, r_t)$ *fails on* x if the rule in (a) gives the wrong answer on x. By (a), a choice can only fail on inputs $x \in L$, and for each fixed such x a random choice fails with probability at most $2^{-(n+1)}$. There are at most 2^n such inputs of length n, so by the union bound a random choice fails on at least one input with probability at most $2^n \cdot 2^{-(n+1)} = 1/2$. Hence at least half of all choices fail on no input of length n, and any such choice has the required property.

(c) For each n fix strings $r_1, \ldots, r_{n+1}$ as in (b), and let $T(n)$ be the (polynomial) running time of M on inputs of length n. By the standard simulation of Turing machines by circuits (the proof that $\mathsf{P} \subseteq \mathsf{P/poly}$), for each i there is a circuit of size $O(T(n)^2)$ that computes $x \mapsto M(x, r_i)$, with r_i hardwired. Let C_n be the OR of these $n+1$ circuits. By (b), C_n decides L on inputs of length n, and its size is $O(n\,T(n)^2)$, which is polynomial. What is hardwired is the $n+1$ strings, that is, $(n+1)\,p(n)$ bits.

(d) Now M has two-sided error: $\Pr_r[M(x,r) = L(x)] \geq 2/3$ for every x. Combine the t runs by *majority* instead of OR. Let X_i indicate that $M(x,r_i) = L(x)$. The majority is wrong only if $\sum_i X_i \leq t/2$, which by the Chernoff bound has probability at most $e^{-t/18}$. Take $t = 18n$, so that this is at most $e^{-n} < 2^{-n}$. Now a choice of strings can fail on any input, not only on those in L, but there are only 2^n inputs of length n, so a random choice fails on at least one of them with probability less than $2^n \cdot 2^{-n} = 1$. Hence some fixed choice $r_1, \ldots, r_t$ is correct on every input of length n. The circuit C_n is the majority of the $t = 18n$ circuits computing $x \mapsto M(x, r_i)$, and has polynomial size as in (c). See also Arora/Barak Section 7.5.

(2) Prove that if SAT $\in$ BPP, then SAT $\in$ RP. Your algorithm should use the BPP decider to construct an assignment and should never accept an unsatisfiable formula. Bound the error and running time, including the cost of amplification.

Solution. Let A be a polynomial-time randomized algorithm that decides SAT with error at most $1/3$ on every input, and let φ be a CNF formula with variables $x_1, \ldots, x_v$, $v \geq 1$.

Amplification. Let A' run A independently $k = \lceil 18 \ln(3v) \rceil$ times and output the majority answer (ties count as NO). By the Chernoff bound in Problem (1)(d), A' errs on every input with probability at most $e^{-k/18} \leq 1/(3v)$.

Algorithm. Set $\varphi_0 = \varphi$. For $i = 1, \ldots, v$, run A' with fresh random bits on $\varphi_{i-1} \wedge (\neg x_i)$. If it answers YES, set $a_i = 0$ and $\varphi_i = \varphi_{i-1} \wedge (\neg x_i)$; otherwise set $a_i = 1$ and $\varphi_i = \varphi_{i-1} \wedge (x_i)$. Finally, accept if and only if the assignment $a = (a_1, \ldots, a_v)$ satisfies φ.

No false accepts. The algorithm accepts only when a satisfies φ, which it checks deterministically. So an unsatisfiable φ is accepted with probability 0, whatever answers A' gives.

Satisfiable formulas. Suppose φ is satisfiable. We first claim that if every call to A' answers correctly, then every φ_i is satisfiable. This holds for φ_0. If φ_{i-1} is satisfiable and $\varphi_{i-1} \wedge (\neg x_i)$ is satisfiable, a correct call answers YES and φ_i is that formula. If $\varphi_{i-1} \wedge (\neg x_i)$ is not satisfiable, then $\varphi_{i-1} \wedge (x_i)$ must be, a correct call answers NO, and φ_i is that formula. So φ_v is satisfiable. Its unit clauses force the assignment a, so a satisfies φ and the algorithm accepts.

It remains to bound the probability that some call is wrong. Let E_i be the event that the i-th call is wrong. Once the outcomes of the first $i-1$ calls are fixed, the i-th query is a fixed formula and A' uses fresh random bits, so E_i has conditional probability at most $1/(3v)$. Hence $\Pr[E_i] \leq 1/(3v)$, and by the union bound some call is wrong with probability at most $1/3$. So a satisfiable φ is accepted with probability at least $2/3 \geq 1/2$.

Running time. There are v calls to A', each consisting of $O(\log v)$ runs of A on a formula of size at most $|\varphi| + O(v)$, followed by one evaluation of φ. This is polynomial in $|\varphi|$.

Remark. Since every language in NP reduces to SAT and RP $\subseteq$ NP, this shows that NP $\subseteq$ BPP implies NP = RP.

(3) You may assume the following consequence of the PCP theorem: for some fixed constant $0 < \gamma < 1$, there is a polynomial-time map from a SAT instance x to a 3-CNF ψ_x with $m \geq 1$ clauses such that

$$\begin{aligned} x \in \text{SAT} &: \text{OPT}(\psi_x) = m, \\ x \notin \text{SAT} &: \text{OPT}(\psi_x) \leq (1-\gamma)m. \end{aligned}$$

(a) Show that a deterministic polynomial-time algorithm always outputting an assignment of value at least $(1 - \gamma/2)\text{OPT}$ would imply P = NP.

(b) Suppose the approximation algorithm is randomized and produces such an assignment with probability at least 2/3 on every input. It always outputs an assignment, but may perform badly on unsuccessful runs. Prove that the existence of such an algorithm would imply $\mathsf{NP} = \mathsf{RP}$, not merely $\mathsf{NP} \subseteq \mathsf{BPP}$.

Solution. Evaluate the returned assignment and accept the original SAT instance iff it satisfies more than $(1-\gamma)m$ clauses of ψ_x. On a YES instance, the guarantee gives at least $(1-\gamma/2)m > (1-\gamma)m$ satisfied clauses. On a NO instance, *no* assignment exceeds the threshold. Thus in (a) this is a deterministic SAT decider.

The same test in (b) never accepts a NO instance, even when the approximation algorithm fails to meet its guarantee: its feasible assignment cannot exceed the true optimum. On a YES instance it accepts with probability at least 2/3. The output's value is efficiently checkable, so this is an RP algorithm for SAT. Polynomial-time reductions give $\mathsf{NP} \subseteq \mathsf{RP}$, while an accepting random tape proves $\mathsf{RP} \subseteq \mathsf{NP}$.

(4) Recall the standard reduction from 3SAT to Independent Set: given a 3-CNF ψ with m clauses, the graph G_ψ has one vertex for each occurrence of a literal in a clause, with an edge between two vertices if they belong to the same clause, or if one is an occurrence of a variable and the other an occurrence of its negation.

(a) Show that the size of a maximum independent set in G_ψ is exactly $\mathrm{OPT}(\psi)$, the maximum number of clauses of ψ that can be satisfied simultaneously.

(b) Using the gap from Problem (3), show that a polynomial-time algorithm that on every graph G outputs an independent set of size at least $(1-\gamma/2)$ times the maximum would imply $\mathsf{P} = \mathsf{NP}$.

Solution. (a) Take an assignment satisfying $\mathrm{OPT}(\psi)$ clauses and pick one true literal occurrence from each satisfied clause. No two of these are in the same clause, and no two are contradictory since all are true, so they form an independent set of size $\mathrm{OPT}(\psi)$. Conversely, let S be an independent set. It contains at most one vertex per clause and no contradictory pair, so there is an assignment making every literal in S true, and it satisfies at least $|S|$ clauses. Hence $|S| \leq \mathrm{OPT}(\psi)$.

(b) Given a SAT instance x, compute ψ_x and G_{ψ_x}, run the algorithm, and accept if and only if it returns an independent set of size more than $(1-\gamma)m$. If $x \in$ SAT, then by (a) the maximum is m and the algorithm returns a set of size at least $(1-\gamma/2)m > (1-\gamma)m$. If $x \notin$ SAT, then by (a) every independent set has size at most $(1-\gamma)m$. So this decides SAT in polynomial time.

(5) In the standard definition of PCP, $\mathsf{PCP}(r, q)$, the verifier uses r random coins, and makes q queries to the prover's string Π. What if instead the verifier is allowed to make q adaptive queries, meaning that the verifier first queries the first position of Π, and then based on the value returned, they then can decide on the second position to query, etc. Prove that any language L that has a PCP verifier using r random coins, and q adaptive queries also has a standard (i.e., non-adaptive) verifier using r coins and 2^q queries.

Solution. Fix the input x and the random string ρ. Since each query made by the verifier has value either 0 or 1, the behaviour of the adaptive verifier on (x, ρ) can be described by a binary tree of depth q: each internal vertex is labelled with a position of Π to query, its two

children correspond to the two possible answers, and each leaf is labelled accept or reject. Such a tree has at most $2^q - 1$ internal vertices, so over all possible answers the verifier queries at most 2^q distinct positions.

The non-adaptive verifier, on (x, ρ), computes this tree by simulating the adaptive verifier on every possible sequence of answers (this takes polynomial time whenever 2^q is polynomial, in particular for constant q). It then queries all positions labelling internal vertices at once, follows the path determined by the answers, and outputs the decision at the leaf it reaches. For every x, ρ and Π it accepts if and only if the adaptive verifier does. So it uses the same r coins and has the same completeness and soundness.

(6) Let y and m be integers with $\gcd(y, m) = 1$. We say that y is a quadratic residue mod m if there exists an integer x such that $y \equiv x^2 \pmod{m}$. Give an IP (interactive protocol) for the following language:

$$\text{NonResidue} = \{(y, m) \mid \gcd(y, m) = 1 \text{ and } y \text{ is not a quadratic residue mod } m\}.$$

Solution. Write $\mathbb{Z}_m^*$ for the set of integers in $\{1, \ldots, m-1\}$ that are relatively prime to m. The verifier first checks that $\gcd(y, m) = 1$ and rejects if not. It then picks a uniformly random $r \in \mathbb{Z}_m^*$ (by sampling $r \in \{1, \ldots, m-1\}$ until $\gcd(r, m) = 1$) and a random bit b, and sends

$$z = \begin{cases} r^2 \bmod m & \text{if } b = 0, \\ y\, r^2 \bmod m & \text{if } b = 1. \end{cases}$$

The prover replies with a bit b', and the verifier accepts if and only if $b' = b$.

Completeness. Suppose y is not a quadratic residue. If $b = 0$ then $z = r^2$ is a residue. If $b = 1$ then $z = y\, r^2$ is not a residue: if $y\, r^2 \equiv s^2$ then $y \equiv (s\, r^{-1})^2$, a contradiction. So the prover, who is computationally unbounded, can determine b by checking whether z is a residue, and the verifier accepts with probability 1.

Soundness. Suppose $y \equiv x^2 \pmod{m}$. Then $x \in \mathbb{Z}_m^*$, since any common factor of x and m would divide y. So $y\, r^2 \equiv (xr)^2$, and since $r \mapsto xr$ is a bijection of $\mathbb{Z}_m^*$, the message $y\, r^2$ has exactly the same distribution as r^2. Thus z is independent of b, and any prover answers correctly with probability exactly $1/2$. Running the protocol twice independently and accepting only if both answers are correct gives soundness error $1/4$ while keeping completeness 1.

(7) Explain in one or two sentences why in the proof that $\#\mathrm{SAT} \in \mathsf{IP}$, we need the prime p used in the protocol to be larger than 2^n.

Solution. The protocol does all of its arithmetic mod p, so it only verifies the claimed number of satisfying assignments modulo p. The true number is an integer between 0 and 2^n, where n is the number of variables; when $p > 2^n$, distinct values in this range remain distinct mod p, so verifying the count mod p verifies the count itself. (Soundness also needs p to be large compared with the degrees of the polynomials sent by the prover, but that alone would not require $p > 2^n$.)