COMS 4236: Intro to Computational Complexity — November 2, 2026

Homework 3 Solutions

Instructor: *Toniann Pitassi* — Due: November 12, 2026

1 Question 1

Solution.

(a) If M has full rank, then the map $x \mapsto Mx$ is injective. Therefore, since the prover's reply x' satisfies $Mx' = y$, it must be the case that $x' = x$ and the verifier will accept.

(b) If M is not full rank, there must exist $v \in \mathbb{F}_p^n$ in the kernel (null space) of M. For such v, the vectors $0, v, 2v, 3v, \ldots, (p-1)v \in \mathbb{F}_p^n$ are also in the kernel of M and are all distinct in $\mathbb{F}_p^n$. Therefore, the kernel of M has cardinality at least p.

For any $v \in \ker(M)$, $M(x+v) = Mx$. Therefore, $M^{-1}(y)$ has cardinality at least p. This means that, using the fact that V picks x randomly, the random variable $x \mid Mx$ is a uniform random choice over at least p items. The verifier accepts only if the prover outputs x'. For any distribution over x', $\Pr[x' = (x \mid Mx)] \leq 1/p$.

2 Question 2

Solution.

Since this is PIT over $\mathbb{Q}$, not all polynomials that evaluate to 0 everywhere in F_2 (binary variables) are the zero formal polynomial over $\mathbb{Q}$. For example, the unsatisfiable formula $x \wedge \neg x$ translates to the polynomial $x - x^2$. In $\mathbb{Q}$, this is not the zero formal polynomial. However, it is identically 0 in $\mathbb{F}_2$.
Improving accuracy doesn't help. Even if the PIT subroutine is able to tell whether a polynomial is the zero formal polynomial with perfect accuracy, it would still return that $x - x^2$ is not the zero formal polynomial.

3 Question 3

Solution.

1. In the first step, the honest prover sends $u(x) = g(x,0) + g(x,1)$. Then, $u(0) + u(1) = g(0,0) + g(0,1) + g(1,0) + g(1,1) = H$, so the verifier check in step 2 will pass.

   In the third step, the honest prover will send $v(y) = g(\alpha, y)$. Then, $v(0) + v(1) = g(\alpha,0) + g(\alpha,1) = u(\alpha)$, so the verifier check in the first part of step 4 will pass. Also, $v(\beta) = g(\alpha,\beta)$, so the final verifier check will also pass and the verifier will accept.

2. We begin with the following observation about degree-at-most-one polynomials over $\mathbb{F}_p$. If $k(x), \ell(x)$ are polynomials that are not the same, then the polynomials either match at 0 or 1 points in the field (1 is possible because one of the polynomials may be constant). This follows from the hint that a degree d polynomial has at most d roots.

   Suppose $H \neq g(0,0) + g(0,1) + g(1,0) + g(1,1)$. Since β is drawn uniformly at random (in particular, independently from α and v), the fact that $v(\beta) = g(\alpha, \beta)$ means that $v(y) = g(\alpha, y)$ for all y with probability at least $1 - 1/p$ (using our initial observation).

   Assume this is the case. Then, the fact that $v(0) + v(1) = u(\alpha)$ implies that $g(\alpha, 0) + g(\alpha, 1) = u(\alpha)$. Because α is independent of u, this fact implies that $g(x, 1) + g(x, 0) = u(x)$ for all x with probability at least $1 - 1/p$.

   Assume this is the case as well. Then the fact that $u(0) + u(1) = H$ implies that $H = g(0,0) + g(0,1) + g(1,0) + g(1,1)$. This is a contradiction, so one of our assumptions must be false. By the union bound, the probability that at least one is false is at most $2/p$.

3. The strategy is as follows. If $2\alpha - 1 = 0$, the prover gives up and fails (this only happens with probability $1/p$). Otherwise, the prover will send $u(x) = (\alpha - x)/(2\alpha - 1)$ and $v(y) = 0$.

   Then, $u(0) + u(1) = 1 = H$, so the first check will pass. Also, $u(\alpha) = 0 = v(0) + v(1)$, so the second check will pass. Finally, $v(\beta) = 0 = g(\alpha, \beta)$ for any β.

4 Question 4

Solution.

(a) First, $\mathsf{P} \subseteq \mathsf{PCP}(0, \log n)$ because the verifier can just ignore the proof and decide the language itself. Second, $\mathsf{PCP}(0, \log n) \subseteq \mathsf{P}$ because if the verifier only accesses $O(\log n)$ bits, chosen deterministically, a polytime algorithm can simulate the verifier on every possible setting of those bits and accept if any setting causes the verifier to accept. This gives the correct solution because the verifier is deterministic.

(b) If $L \in \mathsf{NP}$, take the proof to be the NP witness and have the verifier read it all and run the NP verifier. If $L \in \mathsf{PCP}(0, \text{poly}(n))$, take the witness to be the proof and have the verifier run the PCP verifier.

(c) If $L \in \mathsf{PCP}(\text{poly}(n), 0)$, then an exponential time algorithm to decide L is to try all possible random inputs for the verifier and output the majority answer (the verifier ignores the proof so we don't have to make a proof).

5 Question 5

Solution.

1. The algorithm is to pick an arbitrary edge and continue adding edges until there are no more vertex-disjoint edges. This takes at most $O(|E|^2|V|)$ time, so it is polynomial.

   We know there is a vertex cover of size $\tau(G)$. Every edge in the maximal matching has at least

one of its endpoints in the vertex cover. Because the edges are all vertex-disjoint, none of these vertices are reused, so $|S| \leq 2\tau(G)$.

2. Define a graph on vertices $\{1, \ldots, 2n+1\}$ as follows. Vertices v_i and v_{i+1} are connected for $i = 1, 3, 5, \ldots$. Vertex v_{2n+1} has no neighbors. Then, the maximal matching algorithm will find the matching containing all vertices except v_{2n+1}, so $V \setminus S = \{v_{2n+1}\}$. However, the size of the maximum independent set is $n+1$ (take the odd vertices).

   This happens because

   $$\frac{|V| - 2\tau(G)}{|V| - \tau(G)}$$

   need not be a constant.