6.541/18.405 Problem Set 3

due on **Thursday, May 2, 11:59pm**

Rules: You may discuss homework problems with other students and you may work in groups, but we require that you *try to solve the problems by yourself before discussing them with others.* Think about all the problems on your own and do your best to solve them, before starting a collaboration. If you work in a group, include the names of the other people in the group in your written solution. **Write up your *own* solution to every problem**; don't copy answers from another student or any other source. Cite **all** references that you use in a solution (books, papers, people, websites, etc) at the end of each solution.

We encourage you to use LaTeX, to compose your solutions. The source of this file is also available on Piazza, to get you started!

How to submit: Use Gradescope entry code **2P3PEN**.

Please use a separate page for each problem.

Problem 1: Circuit Lower Bounds from Pseudorandom Generators (3 Points)

Question

We showed that circuit lower bound implied "good" PRGs. Here is a sort-of converse:

Show that if there is a $O(\log(n))$-seed 1/10-pseudorandom generator computable in $2^{O(m)}$ time on m-bit seeds, then there is an $\varepsilon > 0$ such that $\mathbf{DTIME}[2^{O(n)}] \not\subset \mathbf{SIZE}[2^{\varepsilon n}]$.

Answer

Say there exists a 1/10 $O(\log(n))$ seed pseudorandom generator $g : \{0,1\}^* \to \{0,1\}^*$, computable by a Turing machine G in 2^{km} time on m-bit seeds. Say specifically (WLOG) g is defined on all x values of length $c\log(n)$ for $n \in \mathbb{N}$, and $|x| = c\log(n) \implies |g(x)| = n$.

For any string y, let $y_{1:l}$ denote the first l bits of y.

Consider the decision problem $f : \{0,1\}^* \to \{0,1\}$ where

$$f(x) = 1 \iff \exists y \in \{0,1\}^{|x|-1} \text{ s.t. } g(y)_{1:|x|} = x \qquad (*)$$

For inputs x of size n, this can be computed in time $2^{n-1} \cdot 2^{k(n-1)} \in 2^{O(n)}$ by looping over the 2^{n-1} possible strings y, and computing $g(y)$ in time $2^{k(n-1)}$ for each.

Say for contradiction that $f \in \mathbf{SIZE}[2^{n/2}]$. Let C be the circuit of size $2^{n/2}$ that decides this for a given n. Observe that for any $y \in \{0,1\}^{n-1}$, $C(g(y)_{1:n}) = 1$. This is because $g(y)_{1:n}$ is in the range of g on $\{0,1\}^{n-1}$, which is the very property that decision problem f checks, and C is a circuit that computes f. Thus

$$\Pr_{y \in \{0,1\}^{n-1}}[C(g(y))_{1:n} = 1] = 1 \quad (*)$$

However,

$$\Pr_{x \in \{0,1\}^{2^{n-1}}}[C(x_{1:n}) = 1] \leq \frac{1}{2} \quad (**)$$

becuase here, the strings $x_{1:n}$ cover all possible n-long bit-strings, and only half of these can possibly be in the image of g on inputs in $\{0,1\}^{n-1}$.

Statements (*) and (**) contradict that g is a 1/10 PRG, since they mean a circuit C of size $2^{n/2}$ can distinguish between the distribution induced by g on strings of length $n-1 > n/2$, and the actual uniform distribution on strings of size 2^{n-1}.

Problem 2: Randomized Approximate Counting with an NP Oracle (12 pts, 3 for each sub-problem)

Question

We will develop a real-life application of SAT solvers. **Assume $\mathbf{P} = \mathbf{NP}$ in this question.** Let $H_{n,k}$ be a pairwise independent hash family of functions from $\{0,1\}^n$ to $\{0,1\}^k$.

(a) Prove that there is a constant $p \in (0,1)$ and a constant $\varepsilon > 0$ such that for every k and $S \subseteq \{0,1\}^n$,

- if $|S| \leq 2^{k-2}$, then

$$\Pr_{h \in H_{n,k}}[\text{there is an } x \in S \text{ such that } h(x) = 0^k] < p - \varepsilon,$$

and

- if $|S| \geq 2^{k-1}$, then

$$\Pr_{h \in H_{n,k}}[\text{there is an } x \in S \text{ such that } h(x) = 0^k] > p + \varepsilon.$$

(b) Use part (a) to show that that there is a randomized polynomial-time algorithm that approximates #SAT within a factor of 4. More precisely, there is a randomized polynomial-time algorithm that given any Boolean formula F outputs a number K such that $\#\mathsf{SAT}(F)/2 \leq K \leq 2 \cdot (\#\mathsf{SAT}(F))$.

(c) Show that for any constant $\varepsilon > 0$, there is a randomized polynomial-time algorithm that approximates #SAT within a factor of $1 + \varepsilon$. (Hint: Try to modify the given formula F in some natural way that changes the number of SAT assignments, then feed the modification to your algorithm from part (b).)

(d) Show that you can derandomize the algorithm. That is, prove that if $\mathbf{P} = \mathbf{NP}$ then for every function $f \in \#\mathbf{P}$, and any constant $\varepsilon > 0$, there is a deterministic polynomial-time algorithm that approximates f within a factor of $1 + \varepsilon$. (Warning: the approximate counting problem is *not* a decision problem, so you cannot just "plug in" $\mathbf{P} = \mathbf{NP}$ here...)

Answer

Answer to (a)

Say $|S| \leq 2^{k-2}$. For $x \in S$, let A_x denote the event $A_x = \{h : h(x) = 0^k\}$. Then

$$\Pr[\exists x \in S \text{ s.t. } h(x) = 0^k] = \Pr[\cup_{x \in S} A_x] \leq \sum_{i=1}^{|S|} \Pr[A_{x_i}] = \sum_{i=1}^{|S|} \frac{1}{2^k} = |S|/2^k \leq 1/4$$

Now say S' is a set such that $|S'| \geq 2^{k-1}$. Let $S \subseteq S'$ be a subset of size exactly 2^{k-1}. I will show that $\Pr[\cup_{x \in S} A_x] \geq k$ for a certain k. Since $S \subseteq S'$, this will imply $\Pr[\cup_{x \in S'} A_x] \geq k$.

Let $T := \sum_{x \in S} 1_{A_x}$, the number of x values in S such that $h(x) = 0^k$. Then $E[T] = |S|/2^k = 1/2$, and since the indicators 1_{A_x} are pairwise independent random variables,

$$\text{Var}[T] = \sum_{x \in S} \text{Var}[1_{A_x}] = \sum_{x \in S} 2^{-k}(1 - 2^{-k}) = 2^{-1}(1 - 2^{-k}) = \frac{1}{2} - \frac{1}{2^{k+1}}$$

I wish to upper bound $\Pr(\cup_{x \in S} A_x) = \Pr(T \geq 1)$. I proceed as follows, using Cantelli's inequality:

$$\begin{aligned}\Pr(T = 0) = \Pr(T \leq 0) = \Pr(T - \frac{1}{2} \leq -\frac{1}{2}) = \Pr(T - E[T] \leq -\frac{1}{2}) &\leq \frac{\text{Var}[T]}{\text{Var}[T] + (\frac{1}{2})^2} \\ &= \frac{\frac{1}{2} - \frac{1}{2^{k+1}}}{\frac{1}{2} - \frac{1}{2^{k+1}} + 1/4} \leq \frac{1/2}{1/2 + 1/4} = 2/3\end{aligned}$$

Thus,

$$\Pr(\exists x \in S \text{ s.t. } h(x) = 0^k) = \Pr(T \geq 1) = 1 - \Pr(T = 0) \geq 1/3$$

Take $p = \frac{7}{24}$ and $\epsilon = \frac{0.99999}{24}$. Then when $|S| \leq 2^{k-2}$, $\Pr[\exists x \in S.h(x) = 0^k] \leq 1/3 < p - \epsilon$ and when $|S| \geq 2^{k-1}$, $\Pr[\exists x \in S.h(x) = 0^k] \geq 1/4 > p + \epsilon$.

Answer to (b)

Let m be a constant whose value we will set below.

RandomizedApproximateCounting

Require: Formula F on n variables.

1: **for** $k = 0, \ldots, n$ **do**
2: Initialize a count c to 0.
3: $N \leftarrow \lceil (n + m)/\epsilon^2 \rceil$
4: **for** $i = 0, \ldots, N$ **do**
5: Generate a pairwise independent hash function $h : \{0,1\}^n \to \{0,1\}^k$.
6: Construct a formula $\phi \leftarrow (x \mapsto [F(x) \wedge h(x) = 0^k])$.
7: Use a polynomial time SAT solver to compute $s_\phi \leftarrow \mathsf{SAT}(\phi)$.
8: If the $s_\phi = 1$, increment c.
9: **end for**
10: If $c/N < p$, return 2^{k-2}.
11: **end for**
12: Return n.

The algorithm RandomizedApproximateCounting returns a value 2^k which, with probability $> 2/3$, approximates $\#\mathsf{SAT}(F)$ within a factor of 4.

The proof of this has two parts. First, we must show that if $\#\mathsf{SAT}(F) < 2^{k-1}$, it is very unlikely that the algorithm returns the value 2^k. Second, we must show that if $\#\mathsf{SAT}(F) > 2^{k+1}$, it is very likely that the algorithm returns 2^k or a smaller value (in which case it is guaranteed to reach the part of the for loop where it will return a value 2^{k+1} or greater).

Say $\#\mathsf{SAT}(F) \geq 2^{k-1}$ and consider the kth iteration of the outer for loop. By part (a), taking S to be the set of assignments satisfying F, the probability that $s_\phi = 1$ is $\geq p + \epsilon$ at any iteration of the inner loop. Let c_k denote the value of c in the algorithm on iteration k. Then $\Pr[c_k < Np] \leq \Pr_{v \sim \text{Binomial}(p+\epsilon, N)}[v < Np]$. By Hoeffding's inequality, this implies

$$\Pr[c_k < Np] \leq e^{-2N\epsilon^2}$$

Since $N \geq \frac{n+m}{\epsilon^2}$,

$$\Pr[c_k < Np] \leq e^{-2(n+m)}$$

Thus, the probability that the algorithm returns in an iteration k where $2^{k-1} \leq \#\mathsf{SAT}(F)$ is

$$\Pr[\exists k \text{ s.t. } 2^{k-1} \leq \#\mathsf{SAT}(F) \text{ and } c_k < Np] = 1 - \prod_{k=0}^{\log(\#\mathsf{SAT}(F))+1} \Pr[c_k \geq Np]$$

where this equality follows because the result of the algorithm at different iterations of the outer loop are all statistically independent. We can continue bounding this:

$$1 - \prod_{k=0}^{\log(\#\mathsf{SAT}(F))+1} \Pr[c_k \geq Np] \leq 1 - \prod_{k=0}^{n} \Pr[c_k \geq Np]$$

$$= 1 - \prod_{k=0}^{n} (1 - \Pr[c_k < Np]) \leq 1 - (1 - e^{-2(n+m)})^n \leq ne^{-2(n+m)} < 2/3$$

This inequality holds for all n if we choose a sufficiently large m. This guarantees that it is unlikely we return a value $2^{k-2} \leq \frac{1}{2}\#\mathsf{SAT}(F)$.

Now we need to prove that if $\#\mathsf{SAT}(F) \leq 2^{k-2}$ the algorithm has a high probability of returning 2^{k-2}. This means that on the first iteration of the loop with a value k such that $2^{k-2} \geq \#\mathsf{SAT}(F)$, ie. where 2^k is significantly larger than $\#\mathsf{SAT}(F)$, it is almost certain that the loop will return the value 2^{k-2}. This guarantees that it is very unlikely the algorithm returns a value $2^{k-1} \geq 2\#\mathsf{SAT}(F)$ (which is what would occur at the next iteration of the outer loop, if the algorithm failed to detect when $\#\mathsf{SAT}(F) \leq 2^{k-2}$). Combining this with the above result that it is very unlikely the algorithm, returns $2^{k-2} \leq \frac{1}{2}\#\mathsf{SAT}(F)$, we derive that with high probability, the algorithm returns a value $2^{k-2} \in (\frac{1}{2}\#\mathsf{SAT}(F), 2\#\mathsf{SAT}(F))$.

For this part of the proof, note that by part (a), $\#\mathsf{SAT}(F) \leq 2^{k-2}$ implies that $\Pr[s_\phi = 1] > p - \epsilon$ in the algorithm, so by the same analysis as last time using the Hoeffding bound, $\Pr[c_k \geq Np] < e^{-2(n+m)} < 1/3$. (As before, getting this lower bound depends upon choosing a large enough constant , m.) Thus it is highly likely that $c_k < Np$ so the algorithm will return (line 10) in any iteration of the loop where $\#\mathsf{SAT}(F) \leq 2^{k-2}$, if it reaches such an iteration.

Answer to (c)

Let F be a formula on variables $x_1, \ldots, x_n$. For $i = 2, \ldots, n$, let F_i be a formula, identical to F, but on a distinct set of variables $y_1^i, \ldots, y_n^i$.

Let $G_i = F \wedge F_2 \wedge \cdots \wedge F_i$, a formula on ni variables. Observe that if F has s satisfying assignments, G_i has s^i satisfying assignments.

Choose i large enough that $(1+\epsilon)^i > 4$. Let s_i denote the number of satisfying assignments to G_i. Run our algorithm from part (b) on G_i to find a value K such that $K \in [s_i/2, 2s_i]$. Then $s_i \in [K/2, 2K]$. Thus $s^i \in [K/2, 2K]$. Let $k = K/2$, so $s^i \in [k, 4k]$. Then

$$s \in [(k)^{1/i}, 4^{(1/i)}(k^{1/i})] \subseteq [(k)^{1/i}, (1+\epsilon)(k^{1/i})]$$

Thus, this has yielded a $(1+\epsilon)$ approximation to $\#\mathsf{SAT}(F)$.

Answer to (d)

First, observe that if $\mathbf{P} = \mathbf{NP}$, the polynomial hierarchy collapses, so since $\mathbf{BPP} \subseteq \mathbf{PH}$, we have $\mathbf{BPP} \subseteq \mathbf{P}$. I do not know a way to directly use this fact to answer this question, but I will give a derandomization based on a similar idea to how we proved $\mathbf{BPP} \subseteq \mathbf{PH}$.

Consider lines 1-9 in RandomizedApproximateCounting. Let $A(F, k, r)$ denote the Turing machine that runs these lines of the algorithm for loop iteration f, given formula F, invoking randomness r, and outputting the indicator $1_{c/N<p}$. (That is, A outputs 1 iff $c/N < p$ when line 9 of the algorithm is reached.)

In part (b), we proved that If $|S| \leq 2^{k-2}$ then with probability $\geq 2/3$, $A(F, k, r) = 1$, and if $|S| \geq 2^{k-1}$ then with probability $\geq 1/3$, $A(F, k, r) = 0$. By repetition, with $O(n)$ iterates, we can boost these probabilities to $1-2^{-n}$ and 2^{-n}. Let A' be a (still polynomial probabilistic time algorithm) which is boosted to probabilities $1 - 2^{-100n}$, 2^{-100n} for inputs of length n.

Let M_n be the set of n-strings (F, k) such that the resulting set S satisfies $|S| \geq 2^{k-1}$. For any $x \in M_n$, $\Pr_r[A'(x, r) = 1] < 2^{-100n}$. Since $|M_n| \leq 2^n$, Thus $\Pr_r[\exists x \in M_n.A'(x, r) = 0] < 2^n 2^{-100n} = 2^{-99n}$. Let M'_n be the set of n-strings (F, k) such that the resulting set S satisfies $|S| \leq 2^{k-2}$. By the same logic, $\Pr_r[\exists x \in M'_n.A'(x, r) = 0] < 2^{-99n}$. Thus

$$\Pr_r[\exists x \in M'_n.A'(x, r) = 0 \vee \exists x \in M_n.A(x, r) = 1] < 2^{-99n+1} < 1$$

Thus there is some random string r^* such that

$$\forall x \in M'_n.A'(x, r^*) = 1 \wedge \forall x \in M_n.A'(x, r^*) = 0 \quad (*)$$

If $\mathbf{P} = \mathbf{NP}$ (and hence $\mathbf{P} = \mathbf{PH}$), I claim it is possible to in fact construct such a random string r^* in polynomial time. For bitstring y, let T_y be the statement "There exists a bitstring z such that (*) holds with $r^* := y \cdot z$." Since we can upper bound the length of r^*, the quantification for z in this statement can be quantification of a fixed length. Thus the statement $T_y \in \mathbf{PH}$ (it requires only 2 levels of quantification), and so can be decided in polynomial time. Thus, we can apply the idea underlying the SearchSAT to SAT reduction to find a string r^* with property (*) in polynomial time. (We first check if such an r^* exists where the first bit is 1, then set the first bit accordingly. Then we find the second bit, and so on.)

Let B be the deterministic polynomial time algorithm which on input (F, k) first finds a string r^* such that satisfies property (*), and then runs $A'(F, k, r^*)$ and returns the result. Algorithm B is a deterministic algorithm which yields 1 on any (F, k) such that $|S| \leq 2^{k-2}$ and which yields 0 on any (F, k) such that $|S| \geq 2^{k-1}$.

Thus, if we replace lines 1-9 in RandomizedApproximateCounting with algorithm B, this yields a derandomized algorithm which is also guaranteed to return a multiplicative 4-approximation to #SAT. The same boosting argument from part (c) can then be applied to boost this to an arbitrary $(1 + \epsilon)$ approximation.

Problem 3: Constant Round Arthur-Merlin Collapses (3 points)

Question

Prove that for every fixed positive integer k, $\mathbf{AM}[k] \subseteq \mathbf{AM}[2]$.

Hint: Try error-reduction, to make the probability of error very small.

Answer

WLOG assume k is even. Consider an AM[k] protocol to decide $f(x)$. Say on any iteration, a sequence of messages $r_1, m_1, r_2, m_2, \ldots, r_{k/2}, m_{k/2}$ are sent, where r_i are the random messages from the verifier, and m_i are the messages from the prover. Fix a given prover P. Let A_{acc} denote the event that the r_i result in V accepting, and let A_{rej} denote the event that the r_i result in V rejecting. By the definition of an AM protocol, if $f(x) = 1$ there is a prover so $\Pr[A_{\text{acc}}] > 2/3$ and if $f(x) = 0$ then for every prover, $\Pr[A_{\text{rej}}] < 1/3$.

By the error reduction lemma, there is a manner in which this protocol can be run in parallel $O(k)$ times that produces an AM[k] protocol such that if $f(x) = 1$, there is a prover so $\Pr[A_{\text{acc}}] > 1 - 2^{-k}$, and if $f(x) = 0$, for every prover, $\Pr[A_{\text{rej}}] > 1 - 2^{-k}$.

Now, consider the following $\mathbf{AM}[2]$ protocol. On the first round, the verifier sends a sequence of random bits $r_1, r_2, \ldots, r_{k/2}$, where each sequence of bits is as long as the longest sequence that r_i could have been in the $\mathbf{AM}[k]$ protocol with exponential error bounds. The prover will send a message which is a concatenation $m_1, m_2, \ldots, m_{k/2}$, and the verifier will accept if $r_1, m_1, r_2, m_2, \ldots, m_{k/2}$ would have been an acceptable transcript in the $\mathbf{AM}[k]$ protocol with exponential error bounds.

Analysis. If $f(x) = 1$, then there is a prover which would have almost certainly been accepted in the $\mathbf{AM}[k]$ version of the protocol, and running it in this $\mathbf{AM}[2]$ will succeed with equally high probability. Now say $f(x) = 0$. Let r denote the full string of randomness sent by the verifier and let m denote the prover's full response. It is sufficient to upper-bound $\Pr_r[\exists m.V(x, m, r) = 1]$. By the union bound,

$$\Pr_r[\exists m.V(x, m, r) = 1] \leq \sum_m \Pr_r[V(x, m, r) = 1] \leq \sum_m 2^{-k} \leq 2^M / 2^k$$

where M is an upper bound on the length of m (which is therefore upper-bounded by a polynomial in the input size), and k is the value we chose in the error bound for the reduced-error $\mathbf{AM}[k]$ protocol. If we choose to set $k > M + 2$, we get

$$\Pr_r[\exists m.V(x, m, r) = 1] < 1/4$$

which is certainly sufficient.

Problem 4: AM Protocol for Set Lower Bound (6 Points, 2 for each sub-problem)

Question

In this problem, we will develop an **AM** protocol for proving a set lower bound, which is used as a subroutine in the **AM** protocol for graph non-isomorphism. In a set lower bound protocol, the prover needs to prove to the verifier that given a (large) set $S \subseteq \{0,1\}^m$ (where membership in S is efficiently verifiable), S has cardinality at least K, up to a factor of 2. More precisely, given any K,

- if $|S| \geq K$ then the prover can make the verifier accept with high probability;
- if $|S| < K/2$ then the verifier rejects with high probability regardless of what the prover does.

(a) Let $H_{m,k}$ be a pairwise independent hash family of functions from $\{0,1\}^m$ to $\{0,1\}^k$. Use the pairwise independent hash family $H_{m,k}$ to give a 2-round **AM** protocol for the set lower bound problem described above.

(b) Show that there exists an **AM** protocol for set lower bound with perfect completeness.

Hint: Consider the case where the prover uses multiple hash functions $h_1, \ldots, h_n$ so that $\bigcup_{i=1}^{n} h_i(S) = \{0,1\}^k$.

(c) Generalize the idea from part (b) to show that every problem in **MA** has a protocol with perfect completeness. Namely, show that for every language $L \in \mathbf{MA}$, there exists a probabilistic polynomial time verifier V such that

- If $x \in L$, then there exists m such that $\Pr_r[V(x,r,m) = 1] = 1$.
- If $x \notin L$, then for all m, $\Pr_r[V(x,r,m)] \leq 1/3$.

Answer to (a)

The protocol. Choose k so that $K \in [2^{k-2}, 2^{k-1}]$. Let $\mathcal{H}$ be a family of pairwise independent hash functions from $\{0,1\}^m$ to $\{0,1\}^k$. Let $N = 600$. The verifier sends N random hash function $h_1, \ldots, h_M$ to the prover, and N independent values chosen uniformly at random from $\{0,1\}^k$, $y_1, \ldots, y_N$. For each $i = 1, \ldots, N$, the prover will either say that there is no $x \in S$ so that $h_i(x) = y_i$, or it will send a value $x_i \in S$ such that $h_i(x_i) = y_i$. The verifier will accept iff $h_i(x_i) = y_i$ for all i where an x_i was sent, and the number m of values x_i that the prover sent satisfies

$$m > \frac{7}{6}\frac{K}{2^{k+1}}N$$

Below we will define $p := \frac{7}{6}\frac{K}{2^{k+1}}$, so we can write this condition as $m > Np$.

Analysis. Let $A_y := \{h : \exists x \in S.h(x) = y\}$. Recall $K \in [2^{k-2}, 2^{k-1}]$. If $|S| \leq K/2$,

$$\Pr[A_y] \leq \sum_{x \in S} \Pr_h[h(x) = y] = \frac{|S|}{2^k} \leq \frac{K}{2^{k+1}}$$

To analyze the case $|S| \geq K$, I will use Cantelli's inequality again. To lower bound $\Pr[A_y]$ in this case, it suffices to lower bound $\Pr[A_y]$ assuming $|S| = K$, since $\Pr[A_y]$ is nondecreasing in $|S|$. Henceforth assume $|S| = K$. Let $A_{x,y} := \{h : h(x) = y\}$. Observe $A_y = \cup_x A_{x,y}$. Let $T = \sum_{x \in S} 1_{A_{x,y}}$. Observe $E[T] = K/2^k$. Further,

$$\mathrm{Var}[T] = \sum_{x \in S} \mathrm{Var}[1_{A_{x,y}}] = K\mathrm{Var}[1_{A_{x,y}}] = K(2^{-k}(1 - 2^{-k})) = \frac{K}{2^k}(1 - 2^{-k})$$

The first equality follows from the decomposition of variance for pairwise independent random variables (here, the $1_{A_{x,y}}$), and the third equality follows from the fact that $1_{A_{x,y}}$ is a Bernoulli(2^{-k}) random variable.

We now apply Cantelli's inequality to lower bound $\Pr[A_y] = \Pr[T \geq 1] = 1 - \Pr[T = 0]$. By Cantelli's inequality,

$$\begin{aligned}\Pr[T = 0] = \Pr[T \leq 0] &= \Pr[T - \frac{K}{2^k} \leq -\frac{K}{2^k}] = \Pr[T - E[T] \leq -K/2^k] \\ &\leq \frac{\mathrm{Var}[T]}{\mathrm{Var}[T] + (\frac{K}{2^k})^2} = \frac{\frac{K}{2^k}(1 - 2^{-k})}{\frac{K}{2^k}(1 - 2^{-k}) + \frac{K}{2^k} \cdot \frac{K}{2^k}} = \frac{1 - 2^{-k}}{1 - 2^{-k} + K2^{-k}} = \frac{2^k - 1}{2^k - 1 + K}\end{aligned}$$

Thus

$$\Pr[A_y] - 1 - \Pr[T = 0] \geq 1 - \frac{2^k - 1}{2^k - 1 + K} = \frac{K}{2^k + K - 1}$$

Now, plugging in $K \leq 2^{k-1}$,

$$\Pr[A_y] \geq \frac{K}{2^k + K - 1} \geq \frac{K}{2^k + 2^{k-1} - 1} = \frac{K}{3 \cdot 2^{k-1} - 1} = \frac{4}{3}\frac{K}{2^{k+1}} \times \frac{3 \cdot 2^{k-1}}{3 \cdot 2^{k-1} - 1} \geq \frac{4}{3}\frac{K}{2^{k+1}}$$

Thus if $p = \frac{7}{6}\frac{K}{2^{k+1}}$ and $\epsilon = \frac{0.9999}{6}\frac{K}{2^{k+1}}$,

$$|S| \leq K/2 \implies \Pr[A_y] < p - \epsilon$$

and

$$|S| \geq K \implies \Pr[A_y] > p + \epsilon$$

Thus, by the Hoeffding bound, if $|S| \geq K$, $\Pr[m > pN] \leq e^{-2N\epsilon^2}$. So if we take $N \geq \frac{1}{\epsilon^2}$, $\Pr[m > pN] \leq e^{-2} < 1/3$. Likewise, if $|S| \leq K/2$, $\Pr[m \leq pN] < 1/3$.

Finally, notice that since $\frac{0.9999}{6}\frac{K}{2^{k+1}} \leq \frac{0.9999}{6}\frac{2^{k-1}}{2^{k+1}} = \frac{0.999}{24}$, it suffices to take $N = 600$.

Answer to (b)

The protocol. Let n be an integer such that

$$\frac{4(n\log(K)+2)}{2^{n+1}} < \frac{1}{100}$$

Let k be such that $K^n \in [2^{k-2}, 2^{k-1}]$. Let S^n denote the set $S^n = \{x_1 \cdot x_2 \cdot \cdots \cdot x_n : \forall i. x_i \in S\}$. In the protocol, the prover will first send $l = 4k$ hash functions, $h_1, \ldots, h_l$, each s.t. $h_i : \{0,1\}^{nm} \to \{0,1\}^k$, to the verifier. The prover is claiming that for any $y \in \{0,1\}^k$, there exists an i and a $x \in S^n$ s.t. $h_i(x) = y$. The verifier then sends one uniformly random string $y \in \{0,1\}^k$ to the prover. The prover sends back a pair (i, x) with $i \in \{1, \ldots, l\}$ and $x \in S^n$. If $h_i(x) = y$, the verifier accepts; otherwise it rejects.

Analysis.

First, say $|S| \le K/2$. Then $|S^n| \le K^n/2^n$. Thus for any i, $|h_i(S^n)| \le K^n/2^n$. Thus

$$|\cup_{i=1}^{l} h_i(S^n)| \le \sum_{i=1}^{l} |h_i(S^n)| \le \sum_{i=1}^{l} K^n/2^n = lK^n/2^n$$

Thus

$$\frac{1}{2^k}|\cup_{i=1}^{l} h_i(S^n)| \le \frac{lK^n}{2^{n+k}} = \frac{4k}{2^n}\frac{K^n}{2^k} \le \frac{4k}{2^n}\frac{2^{k-1}}{2^k} = \frac{4k}{2^{n+1}}$$

Observe that since $K^n \ge 2^{k-2}$, $k-2 \le \log(K^n)$ so $k \le \log(K^n)+2$.

Thus

$$\frac{1}{2^k}|\cup_{i=1}^{l} h_i(S^n)| \le 4(n\log(K)+2)/2^{n+1} < 1/100$$

where the last inequality follows due to our choice of sufficiently large n.

This shows that the fraction of the values in $\{0,1\}^k$ covered by the sets $h_i(S)$ is les than $1/100$. Thus, regardless of what l hash functions the prover sends to the verifier, if $|S| \le K/2$, with probability $99/100$, the prover will not be able to produce an (i, x) pair such that $h_i(x) = y$ for the random $y \in \{0,1\}^k$ the verifier sends to the prover.

Now, say $|S| \ge K$. To show that the protocol has perfect completeness, our goal is to show that in this case, there exist some hash functions $h_1, \ldots, h_n$ such that $\forall y \in \{0,1\}^k, \exists i \in \mathbb{Z}_l, x \in S^n . h_i(x) = y$. Then, the prover can simply send these hash functions, and is guaranteed to be able to respond successfully to any query the verifier makes (that is, any y it sends.) To prove this I will use the probabilistic method, and show that if the hash functions are generated uniformly at random, there is nonzero probability of obtaining a set $h_{1:l} := \{h_1, \ldots, h_l\}$ satisfying this condition.

Let $A_{i,y} := \{h_{1:l} : \exists x. h_i(x) = y\}$. The goal is to show that

$$\Pr_{h_{1:l}}[\cap_y \cup_i A_{i,y}] > 0$$

First, letting $A^C_{i,y}$ denote the complement of $A_{i,y}$, note that

$$\Pr[\cap_y \cup_i A_{i,y}] = 1 - \Pr[\cup_y \cap_i A^C_{i,y}] \ge 1 - \sum_{y\in\{0,1\}^k} \Pr[\cap_i A^C_{i,y}] = 1 - 2^k \Pr[\cap_i A^C_{i,0^k}] \quad (*)$$

Because the h_i are independent, $\Pr[\cap_i A^C_{i,0^k}] = \Pr[A^C_{1,0^k}]^l$. Our analysis from part (a) shows that $\Pr[A_{1,0^k}] \ge \frac{4}{3}\frac{K}{2^{k+1}}$ and because $K \ge 2^{k-2}$, this implies $\Pr[A_{1,0^k}] \ge \frac{4}{3}\frac{1}{8} = \frac{1}{6}$. Thus $\Pr[A^C_{1,0^k}] \le \frac{5}{6}$, so $\Pr[\cap_i A^C_{i,0^k}] \le (\frac{5}{6})^l$. Combining this with (*), we see

$$\Pr[\cap_y \cup_i A_{i,y}] \ge 1 - 2^k \Pr[\cap_i A^C_{i,0^k}] \ge 1 - 2^k(\frac{5}{6})^l = 1 - 2^k(5/6)^{4k} > 1 - 2^k(1/2)^k = 0$$

The second inequality here plugged in $l = 4k$, which we set at the beginning of the protocol. We chose $4k$ specifically because $(5/6)^4 < 1/2$.

This strict inequality $\Pr[\cap_y \cup_i A_{i,y}] > 0$ completes the proof.

Answer to (c)

Consider a decision problem $f : \{0,1\}^* \to \{0,1\}$ decided by a Merlin-Arthur protocol with verifier V. WLOG, say that the transcripts for this protocol use exactly kn^k random bits on inputs of size n. Fix an input x and an optimal prover P. Let $n = |x|$ and let $R = kn^k$. Let $S \subseteq \{0,1\}^R$ denote the set of random strings on which (V, P) will accept x. It is guaranteed that either $|S| \geq \frac{2}{3}2^R$ and $f(x) = 1$, or $|S| \leq \frac{1}{2}2^R$ and $f(x) = 0$ (this is the "BPP promise" for the interactive proof protocol). We can slightly adapt the idea from part (b) to have the prover prove to the verifier, with perfect completeness, that $|S| \geq \frac{2}{3}2^R$.

The protocol. Identically to as in the protocol in part (b), the prover first sends l hash functions $h_1, \ldots, h_l$. The verifier then sends a random value $y \in \{0,1\}^t$ where here t plays the role played by k in part (b) [it is computed analogously to there]. The prover then sends back a string $x_1 \cdot x_2 \ldots x_M \in \{0,1\}^{MR}$ for some multiplier M (called n in part (b)), and an integer i, claiming that (1) $h_i(x_1, \ldots, x_M) = y$ and (2) each x_j is in the set S. The verifier can immediately verify condition (1). To enable the verifier to confirm condition (2), the prover simply needs to send the verifier the messages it would have sent in a round of interaction where the verifier sent the string x_j; the verifier can then check that it would have accepted on this response.

Analysis. Per the analysis in part (b), this procedure has perfect completeness. To show soundness all that we need to do is ensure that the verifier can succeed in checking whether $x_j \in S$. Well, say $x_j \notin S$. Due the assumed optimality of P, this implies there does not exist a response to string x_j which would have caused the verifier to accept. Thus, the prover in the protocol in the previous paragraph could not have produced a message to convince the verifier $x_j \in S$. (This shows that in the protocol above, the verifier never mistakenly thinks that a string $x_j \in S$. It does not show the protocol has perfect soundness, for the same reason as in part (b): there is a small but nonzero chance that the verifier happens to choose a string y where some sequence $x_1 \ldots x_M$ of values in S hash to y under one of the h_i.)

Problem 5: The Limits of PCPs (4 Points, 2 for each sub-problem)

Question

Recall that in class we defined $\mathbf{PCP}_s[r(n), q(n)]$ to be the set of functions with probabilistically checkable proofs having "soundness" s. In general, we can parametrize the "completeness" as well.

Specifically, define $f : \{0,1\}^* \to \{0,1\}$ to be in $\mathbf{PCP}_{c,s}[r(n), q(n)]$ if there is a probabilistic polynomial time algorithm V such that for all x, V uses $O(r(|x|))$ random bits, asks $q(|x|)$ oracle queries to a proof string P non-adaptively, must decide whether accept or reject, and

- $f(x) = 1 \implies$ there is a P such that $\Pr[V^P(x) \text{ accepts}] \geq c$.
- $f(x) = 0 \implies$ for all P, $\Pr[V^P(x) \text{ accepts}] < s$.

Note that in this generalized version, when $f(x) = 1$, we do not require the verifier to accept with probability 1 on some proof P.

In the PCP lectures, it was proved that $\mathbf{PCP}_{1,1}(\log n, 3) = \mathbf{NP}$. The number 3 here is actually the smallest possible. In this problem, you are asked to show that if we reduce the number of queries to two or one, the classes become $\mathbf{P}$. Prove that:

(a) for every $0 < s \leq c \leq 1$, $\mathbf{PCP}_{c,s}(\log n, 1) = \mathbf{P}$.

(b) for every $0 < s \leq 1$, $\mathbf{PCP}_{1,s}(\log n, 2) = \mathbf{P}$.

Hint: Think about these 1-query and 2-query PCPs from the CSP/inapproximability perspective: what you want to show is that the resulting CSPs are in fact easy to solve.

Extra credit: Prove that for every $0 < s \leq 1$, $\bigcup_{k \geq 1} \mathbf{PCP}_{1,s}(n^k, 2) \subseteq \mathbf{PSPACE}$

Hint: Use the fact that 2SAT is in NL.

Answer to (a)

For contradiction, say $\mathbf{PCP}_{c,s}(\log n, 1) \not\subseteq \mathbf{P}$. Let $f \in \mathbf{PCP}_{c,s}(\log n, 1) \setminus \mathbf{P}$ be a decision problem, decided by verifier V. I will contradict this by constructing a polynomial time Turing machine that decides f. It will operate as follows. Say input x is given. For each $r \in \{0,1\}^{\log n}$, let $y_r = 1$ if $V(x, r, 1) = 1$ and let $y_r = 0$ otherwise. (Each of these can be computed in polytime.) Here I write $V(x, r, y)$ to denote the value returned by verifier V on input x and randomness r, given that it received value y after the one query it makes to the prover (which is a deterministic function of x and r, given the proof).

Observe that if $f(x) = 1$, then $V(x, r, y_r) = 1$ for at least c of the possible r values. If $f(x) = 0$, then $V(x, r, y_r) = 0$ for at least $1 - s$ of the possible r values (and in fact this would be true for any assignment to the y values). Since there are only n distinct y_r values, our algorithm can simply compute $V(x, r, y_r)$ for each $r \in \{0,1\}^{\log n}$, and accept if $\geq cn$ of them are true.

Answer to (b)

In this problem I will write $V(x, r, y, z)$ to be the value that V outputs on input x and randomness r, given that it receives response y and z from the prover on the two queries it makes. Since the queries are non-adaptive, for any verifier, y and z are a function of x and r, given the proof.

Again for contradiction, say $\mathbf{PCP}_{1,s}(\log n, 2) \not\subseteq \mathbf{P}$. Let $f \in \mathbf{PCP}_{1,s}(\log n, 2) \setminus \mathbf{P}$. I will describe a polytime algorithm that decides f.

Fix input string x. We will construct a 2SAT instance on variables $Y_0, \ldots, Y_{n-1}, Z_0, \ldots, Z_{n-1}$. For each $r \in \{0,1\}^{\log n} = \{0, 1, \ldots, n-1\}$, we will construct a clause C_r as follows. Run $V(x, r, y, z)$ on this x and r for each of the 4 permutations of possible values for y, z. Let C_r be a 2SAT formula on variables Y_r, Z_r so that $C_r(y, z) = V(x, r, y, z)$ for all y, z.[1]

If $f(x) = 1$, then for every r, there is an assignment to Y_r, Z_r so that C_r is satisfied (this is because $c = 1$). If $f(x) = 0$, then for every assignment to Y_r, Z_r, $< sn$ of the C_r are satisfied. Let $C = \wedge_r C_r$, which is a conjunction of 2CNF formulae, and is thus a 2CNF formula itself. We can determine whether C is satisfiable in polynomial time by solving a 2SAT problem; if it is satisfiable, then we must have $f(x) = 1$, and if not, we must have $f(x) = 0$.

[1]Here are the details of the construction of C_r. If $V(x, r, y, z)$ is never 1, set $C_r = Y_r \wedge \neg Y_r$. If $V(x, r, y, z) = 1$ only on y^*, z^*, set $C_r = (Y_r = y^*) \wedge (Z_r = z^*)$. If $V(x, r, y, z) = 1$ exactly when $y = 1$, set $C_r = Y_r$; do the analogus thing if it is 1 exactly when $y = 0$ or exactly when $z = 1$ or exatly when $z = 0$. If $V(x, r, y, z) = 1$ exactly when $y = z$ make $C_r = (Y_r \vee \neg Z_r) \wedge (\neg Y_r \vee Z_r)$. If $V(x, r, y, z) = 1$ exactly when $y \neq z$ make $C_r = (Y_r \vee Z_r) \wedge (\neg Y_r \vee \neg Z_r)$. If $V(x, r, y, z) = 1$ in every case except (y^*, z^*), set $C_r = (Y_r \neq y^*) \vee (Z_r \neq z^*)4$. If $V(x, r, y, z) = 1$ on all y, z, set $C_r = \varepsilon$, the empty clause.

Acknowledgements

I went to Zixuan's and Ryan's office hours, and they gave me advice regarding how to solve problems 2 and 4.