

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 L^AT_EX, 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 $\mathbf{DTIME}[2^{O(n)}] \not\subseteq \mathbf{SIZE}[2^{\varepsilon n}]$.

Answer

TODO: clean this up! Currently messy enough there is a small chance my approach doesn't work.

Say there exists such a pseudorandom generator $g : \{0,1\}^* \rightarrow \{0,1\}^*$ computed by Turing machine G running in 2^{km+l} time. WLOG, say that for all n , $|x| = c \log(n) \implies |g(x)| = n$. For any circuit C of size n ,

$$|Pr_{x \in \{0,1\}^{c \log(n)}}[C(g(x)) = 1] - Pr_{x \in \{0,1\}^n}[C(x) = 1]| < 1/10$$

The idea is going to be that in time $2^m \times 2^{km} = 2^{(k+1)m}$, a Turing machine can run g on every length m string, and compute $Pr_{x \in \{0,1\}^{c \log(n)}}[C(g(x)) = 1]$ exactly, but no circuit of size 2^m can do this.

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

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

Suppose for contradiction that $f \in \mathbf{SIZE}[2^n]$. Let C be the circuit of size 2^n that decides this for a given n . Say $m = cn$ (so $m = c \log(2^n)$). Observe

$$Pr_{y \in \{0,1\}^m}[C(g(y)_{1:m+1}) = 1] = 1$$

Here, $g(y)_{1:m+1}$ is the first $m+1$ bits of $g(y)$.

However,

$$Pr_{x \in \{0,1\}^{2^n}}[C(x_{1:m+1}) = 1] \leq 1/2$$

because only half of the $m+1$ bit strings can be in the range of a function with domain $\{0,1\}^m$. This contradicts (*). Thus $f \notin \mathbf{SIZE}[2^n]$. But certainly $f \in \mathbf{DTIME}[2^{O(n)}]$, because it is possible to loop over all 2^n y values of length n , and for each one check if $x = g(y)$ in time $2^{O(n)}$. This yields an algorithm for deciding f with total runtime $2^n 2^{O(n)} = 2^{O(n)+n} = 2^{O(n)}$.

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 there is a randomized polynomial-time algorithm that approximates $\#\text{SAT}$ 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 $\#\text{SAT}(F)/2 \leq K \leq 2 \cdot (\#\text{SAT}(F))$.
- (c) Show that for any constant $\varepsilon > 0$, there is a randomized polynomial-time algorithm that approximates $\#\text{SAT}$ 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** $k = 0, \dots, n$ **do**
 - 2: Initialize a count c to 0.
 - 3: $N \leftarrow \lceil (n + m)/\epsilon^2 \rceil$
 - 4: **for** $i = 0, \dots, N$ **do**
 - 5: Generate a pairwise independent hash function $h : \{0, 1\}^n \rightarrow \{0, 1\}^k$.
 - 6: Construct a formula $\phi \leftarrow (x \mapsto [F(x) \wedge h(x) = 0^k])$.
 - 7: Use a polynomial time SAT solver to compute $s_\phi \leftarrow \text{SAT}(\phi)$.
 - 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 $\#\text{SAT}(F)$ within a factor of 4.

The proof of this has two parts. First, we must show that if $\#SAT(F) < 2^{k-1}$, it is very unlikely that the algorithm returns the value 2^k . Second, we must show that if $\#SAT(F) > 2^{k+1}$, 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 $\#SAT(F) \geq 2^{k-1}$ and consider the k th 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 $2^{k-1} \leq \#SAT(F)$ is

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

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:

$$\begin{aligned} 1 - \prod_{k=0}^{\log(\#SAT(F))+1} \Pr[c_k \geq Np] &\leq 1 - \prod_{k=0}^n \Pr[c_k \geq Np] \\ &= 1 - \prod_{k=0}^n (1 - \Pr[c_k < Np]) \leq 1 - (1 - e^{-2(n+m)})^n \leq ne^{-2(n+m)} < 2/3 \end{aligned}$$

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

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

For this part of the proof, note that by part (a), $\#SAT(F) \leq 2^{k-2}$ 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 $\#SAT(F) \leq 2^{k-2}$, if it reaches such an iteration.

Answer to (c)

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

Let $G_i = F \wedge F_2 \wedge \dots \wedge F_i$, 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 $\#SAT(F)$.

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 $\#\text{SAT}$. 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 $\mathbf{AM}[k]$ protocol to decide $f(x)$. Say on any iteration, a sequence of messages $r_1, m_1, r_2, m_2, \dots, r_{k/2}, m_{k/2}$ 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 \mathbf{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, by running this protocol in parallel $O(k)$ times (with a minor modification to be described in a moment), we obtain an $\mathbf{AM}[k]$ 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, $\Pr[A_{\text{rej}}] < 2^{-k}$.

[TODO: do we need to prove this lemma?]

Now, consider the following $\mathbf{AM}[2]$ protocol. On the first round, the verifier sends a sequence of random bits $r_1, r_2, \dots, r_{k/2}$, 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 $m_1, m_2, \dots, m_{k/2}$, and the verifier will accept if $r_1, m_1, r_2, m_2, \dots, m_{k/2}$ 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 , 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, \dots, 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 $2^{k-2} \leq K \leq 2^{k-1}$. Let \mathcal{H} be a family of pairwise independent hash functions from $\{0, 1\}^m$ to $\{0, 1\}^k$. The verifier first sends a random hash function h to the prover, then sends N independent values chosen uniformly at random from $\{0, 1\}^k$, y_1, \dots, y_N . For each $i = 1, \dots, N$, the prover will either say that there is no $x \in S$ so that $h(x) = y_i$, or it will send a value $x_i \in S$ such that $h(x_i) = y_i$.

Analysis. [See notebook.]

Answer to (a) [scrapped!]

The protocol. Let \mathcal{H} be a family of pairwise independent hash functions from $\{0, 1\}^m$ to $\{0, 1\}^k$, where k is chosen so that $K \geq 2^{k-1}$. The prover will first send K , and then send a hash function $h \in \mathcal{H}$, such that $|h(S)| \geq K$. The verifier will send back N random values y_1, \dots, y_N . For each $i = 1, \dots, N$, the prover will either say that there is no $x \in S$ so that $h(x) = y_i$, or it will send a value $x_i \in S$ such that $h(x_i) = y_i$. For each x_i value the prover sends, the verifier will confirm that $x_i \in S$ and $h(x_i) = y_i$ and reject otherwise. If all these tests pass, the verifier will accept iff the prover sent at least $\frac{3}{4} \frac{K}{2^k} N$ values x_i .

This protocol uses more than 2 rounds, but we can turn it into a 2-round protocol using the reduction from question 3.

Analysis. It suffices to prove the following three lemmata:

1. Lemma 1. If $|S| \geq K$, there exists a hash function h such that $|h(S)| \geq K$.
2. Lemma 2. If $|h(S)| \geq K$, then with high probability, there exist $\frac{3}{4} \frac{K}{2^k} N$ values x_i such that $h(x_i) = y_i$.
3. Lemma 3. If $|S| < K/2$, then for any function $h : \{0, 1\}^m \rightarrow \{0, 1\}^k$, with high probability, there exist $\frac{3}{4} \frac{K}{2^k} N$ values x_i such that $h(x_i) = y_i$.

Proof of lemma 1. Say $|S| \geq K$. Let h be a random independent hash function. For any $y \in \{0, 1\}^k$, let $A_y := \{h : \exists x \in S. h(x) = y\}$. Let $A_{y,x} := \{h : h(x) = y\}$, so $A_y = \cup_{x \in S} A_{y,x}$. Then by inclusion-exclusion,

$$\Pr_h[A_y] \geq \sum_{x \in S} \Pr[A_{y,x}] - \sum_{x \in S, x < z \in S} \Pr[A_{y,x} \cap A_{y,z}] = |S| \frac{1}{2^k} - (|S|^2 - |S|) \frac{1}{2^{2k}} \geq |S|/2^k - (|S|/2^k)^2$$

Observing that (A) $x - x^2$ is a decreasing function on $(\frac{1}{2}, 1)$, (B) $|S|/2^k \in (\frac{1}{2}, 1)$ because $|S| \geq K$ and $K/2^k > 1/2$, and (C) $\frac{K}{2^k} \geq \frac{|S|}{2^k}$, we derive

$$\Pr_h[A_y] \geq |S|/2^k - (|S|/2^k)^2 \geq \frac{K}{2^k} - \left(\frac{K}{2^k}\right)^2$$

Thus

$$E\left[\sum_{y \in \{0,1\}^k} 1_{A_y}\right] \geq 2^k \left(\frac{K}{2^k} - \left(\frac{K}{2^k}\right)^2\right) = K - \frac{K^2}{2^k}$$

Thus, for a randomly chosen hash function h , we expect that at least $K - \frac{K^2}{2^k}$ distinct y values in $\{0, 1\}^k$ are covered by $h(S)$. This implies that there must exist some hash function h^* with $h^*(S) \geq K - \frac{K^2}{2^k}$.

Proof of lemma 2.

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 $f : \{0, 1\}^* \rightarrow \{0, 1\}$ 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).

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 $r \in \{0, 1\}^n$, 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 .

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

Fix input string x . We will construct a 2SAT instance on variables $Y_0, \dots, Y_{n-1}, Z_0, \dots, Z_{n-1}$. For each $r \in \{0, 1\}^{\log n} = \{0, 1, \dots, n-1\}$, 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 $C = \bigwedge_r C_r$, which is a conjunction of 2CNF formulae, and is thus a 2CNF formula itself. We can determine whether C is satisfiable in polynomial time (this is just a 2SAT problem); if it is, then we must have $f(x) = 1$, and if not, we must have $f(x) = 0$.

¹Here 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 analogous thing if it is 1 exactly when $y = 0$ or exactly when $z = 1$ or exactly 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 $C_r = (Y_r \neq y^*) \vee (Z_r \neq z^*)$. If $V(x, r, y, z) = 1$ on all y, z , set $C_r = \varepsilon$, the empty clause.

Acknowledgements

I went to Zixuan's office hours, and she helped me understand how to solve problems 2 and 4.