Pseudorandomness **Prof. Salil Vadhan**

Problem Set 2

Assigned: Thu. July. 23, 2015 Chi-Ning Chou

Problem 1. (Schwartz-Zippel lemma.) If $f(x_1, ..., x_n)$ is a nonzero polynomial of degree d over a field $\mathbb{F}$ and $S \subset \mathbb{F}$,then

$$\Pr_{\alpha_1,...,\alpha_n \leftarrow S}[f(\alpha_1, ..., \alpha_n) = 0] \leq \frac{d}{|S|}$$

Solution. Prove by the induction on the number of the variables k.
The induction hypothesis is $\underset{\alpha_1,...,\alpha_k \leftarrow S}{Pr}[f(\alpha_1, ..., \alpha_k) = 0] \leq \frac{d}{|S|}$.
For the base case $k = 1$, f reduce to an univariate polynomial which has at most d roots. Thus, the probability to take a root from set S is at most $\frac{d}{|S|}$ since S contains no more than d roots of f. Suppose the induction hypothesis stands for $k \leq n$, consider $k = n + 1$. For any multivariate polynomial f with $n + 1$ variables and an $n + 1$th input α_{n+1}. We can regard them as: $f(\cdot, ..., \cdot, \alpha_{n+1}) = f_{\alpha_{n+1}}(\cdot, ..., \cdot)$. Now, we can factorize $f_{\alpha_{n+1}}$ with the degree of x_{n+1} in order to use Schwartz-Zippel lemma. Suppose the maximum degree of x_{n+1} in f is k, then we can write

$$f_{\alpha_{n+1}} = \sum_{i=0}^{k} q_i(x_1, ..., x_n)\alpha_{n+1}^i$$

, where q_i are polynomials in $x_1, ..., x_n$. Note that the degree of $f_{\alpha_{n+1}}$ is k. Thus, by induction hypothesis, we have

$$\Pr_{\alpha_1,...,\alpha_n \leftarrow S}[f_{\alpha_{n+1}}(\alpha_1, ..., \alpha_n) = 0] \leq \frac{d-k}{|S|}$$

And if $f_{\alpha_{n+1}}(\alpha_1, ..., \alpha_n) \neq 0$, since we take α_{n+1} arbitrary, we have

$$\begin{aligned}\Pr_{\alpha_{n+1} \leftarrow S}[f_{\alpha_{n+1}}(\alpha_1, ..., \alpha_n) = 0] &= \Pr_{\alpha_{n+1} \leftarrow S}[f(\alpha_1, ..., \alpha_{n+1}) = 0] \\ &\leq \frac{k}{|S|}\end{aligned}$$

To sum up,

$$\begin{aligned}
\Pr_{\alpha_1,...,\alpha_{n+1} \leftarrow S}[f(\alpha_1,...,\alpha_{n+1})=0] &\le \Pr_{\alpha_{n+1} \leftarrow S}\left(\Pr_{\alpha 1,...,\alpha_n \leftarrow S}[f_{\alpha_{n+1}}(\alpha_1,...,\alpha_n)=0]\right) \\
&+ \Pr_{\alpha_1,...,\alpha_n \leftarrow S}\left(\Pr_{\alpha_{n+1} \leftarrow S}[f_{\alpha_{n+1}}(\alpha_1,...,\alpha_n)=0]\right) \\
&\le \frac{d-k}{|S|} + \frac{k}{|S|} = \frac{d}{|S|}
\end{aligned}$$

Intuition (Schwartz-Zippel Lemma)

- The statement is for **ANY** subset of the field or domain.
- Utilize the structure of polynomial. For instance, the ratio of zeros can be upper bounded by the degree$d$. When $d = 1$, we can upper bound the exact number of zeros. When $d > 1$, the bound is for ratio.

Problem 2. (Robustness of the model) Suppose we modify our model of randomized computation to allow the algorithm to obtain a random element of $\{1, ..., m\}$ for any number m whose binary representation it has already computed (as opposed to just allowing to random bits). Show that this would not change the classes **BPP** and **RP**.

Solution. Suppose $2^k + 1 < m \le 2^{k+1} + 1$ for some integer k. Thus, the binary representation of m will be $b_1 b_2 ... b_{k+1}$, $b_i \in \{0, 1\}$. Let X be the random sample from $\{1, ..., m\}$, take $X' = X - 1$ and consider the binary representation of X': $b'_1 b'_2 ... b'_{k+1}$. If b'_1, which is the most significant bit, equals to 1, then throw away this number and sample another random number again. Once $b'_1 = 0$, return $b'_2, ..., b'_{k+1}$ as the random bits to the algorithm.
Since we condition on the case that the random number X' being less than 2^k, the distribution of X' will be uniform in $\{0, ..., 2^k\}$ which means that each b_i is uniform in $\{0, 1\}$. As a result, X' provides k random bits. Note that for a single sample, the algorithm generates at most k random bits.
Moreover, the probability to generate a valid random sequence per sample is $\frac{2^k}{m} \ge \frac{1}{2}$. Namely, the expected random bits generated per sample is greater than $\frac{k}{2}$. Thus, using this randomization model is even more efficient than the binary version. In conclusion, when implementing the algorithm in **BPP** or **RP**, allowing the randomized algorithm sample in $\{1, m\}$ will let the computation problem stay in **BPP** or **RP**.

Intuition (support size of random element)

The number of samples needed from a randomized algorithm is inverse-linear in the support size of the random element. That is, Once $m = o(2^n)$, it won't affect the polynomial complexity classes.

Problem 3. (Zero error versus 1-sided error) Prove that **ZPP** = **RP** $\cap$ **co-RP**.

Solution.

1. ($ZPP \subset$ **RP** $\cap$ **co-RP**)
   Suppose language $L \in$ **ZPP**, $\exists$ probabilistic algorithm A s.t. A decides L in polynomial time. Since A decides L with neither false negative error nor false positive error, $L \in$ **RP** and $\mathbf{L} \in$ **co-RP**. Thus, $L \in$ **RP** $\cap$ **co-RP**.

2. ($ZPP \supset$ **RP** $\cap$ **co-RP**)
   Suppose language $L \in$ **RP** $\cap$ **co-RP**, $\exists$ probabilistic algorithms B and C s.t. B decides L in polynomial time with false negative error less than $\frac{1}{3}$ and C decides L in polynomial time with false positive error less than $\frac{1}{3}$. Construct a new probabilistic algorithm D as follow:

   (a) $\forall x \in \{0,1\}^n$, if $B(x) = 0$, then output 0. If not, go to 2.

   (b) If $C(x) = 1$, then output 1. Otherwise, output 0.

   To check the correctness of algorithm D, if $x \in L$, B has some probability fails to recognize it. However, in the second step, C will successfully recognize x. If $x \notin L$, B will successfully recognize x. Also, while B and C are both probabilistic polynomial time algorithm, D is a probabilistic polynomial time algorithm that correctly decides L. Thus, $L \in$ **ZPP**.

Problem 4. (Polynomial Identity Testing for integer circuits)

1. Show that if N is a nonzero integer and $M \xleftarrow{R} \{1, ..., \log^2 N\}$, then

$$Pr[N \not\equiv 0(\text{mod} M)] = \Omega(1/\log\log N)$$

2. Show that Arithmetic Circuit Identity Testing over $\mathbb{Z}$ is in **co-RP**.

Solution.

1. The idea to show this lower bound is simple: show that there must be a large amount of prime numbers in $\{1, ..., \log^2 N\}$ such that they can not divide N by counting argument. Observe that we have the following facts:

   - Number of primes in $\{1, ..., \log^2 N\} = \Theta(\frac{\log^2 N}{\log \log^2 N}) = \Theta(\frac{\log^2 N}{2 \log \log N})$.
   - Number of prime divisors of $N = O(\log N) = o(\frac{\log^2 N}{2 \log \log N})$.

   As the order of the number of primes in $\{1, ..., \log^2 N\}$ is asymptotically greater than the number of prime divisors of N, we can see that there must be a large portion of primes in $\{1, ..., \log^2 N\}$, say half of them. As a result, as N becomes large enough, we have at least $\Theta(\frac{\log^2 N}{4 \log \log N})$ numbers in $\{1, ..., \log^2 N\}$ that cannot divide N. This completes the lower bound:

$$Pr[N \not\equiv 0(\text{mod } M)] = \Omega(1/\log\log N)$$

   , where $M \xleftarrow{R} \{1, ..., \log^2 N\}$.

2. The main difficulty here is that the arithmetic circuit is over $\mathbb{Z}$, which means that as there are polynomially many operations, the value of the output might becomes exponentially large and takes exponential time to compute. To be clear, a single arithmetic operation in arithmetic circuit takes $O(\log A)$ time, where A is the largest number in the whole evaluation. Moreover, if the circuit size if $m = poly(n)$, where n is the input size, the output might be as large as n^{2^m}. As a result, we may want to modulo the result of each operation in order to bound single operation time.

   However, after using modulo to save time, we can see that some nonzero results might be detected as a zero after modulation once the divisor we chose is not good enough. As a result, we also need to show that this false alarm probability can be controlled.

   Consider the following scenario:

   - There are n input and the size of circuit is $m = poly(n)$, thus the maximum degree of the function is less than 2^m.
   - Randomly choose each input from $\{1, ..., 3 \cdot 2^m\}$, which consumes $O(nm)$ random bits. By Schwartz-Zippel lemma the probability of false alarm is less than $\frac{2^m}{3 \cdot 2^m} = \frac{1}{3}$.
   - The largest value of output might be $N = O((3 \cdot 2^m)^{2^m})$.
   - To make sure the value in the whole evaluation process is small, we fo modulation after every operation. The divisor is randomly chosen from $\{1, ..., \log^2 N\}$. We can compute the probability of false alarm by 1., which shows that it is less than $O(1 - \frac{1}{\log\log N}) = O(1 - \frac{1}{m \log m}) = O(e^{-\frac{1}{m \log m}})$.
   - If we do the whole process m^2 times, the false alarm probability will be upper bounded by $O(e^{-\frac{m}{\log m}})$, which is negligible in n.

   To sum up, we can see that all of the steps in the above process take polynomial time, which means that is is indeed a probabilistic polynomial time algorithm. Also, by Schwartz-Zippel lemma and the argument in the last two steps, we can upper bound the false alarm probability. As miss detection cannot happen, we reach a **co-RP** algorithm. Thus, Arithmetic Circuit Identity Testing over $\mathbb{Z}$ is in **co-RP**.

Problem 6. (Primality Testing)

1. Show that for every positive integer n, the polynomial identity $(x+1)^n \equiv x^n + 1 \pmod{n}$ holds iff n is prime.

Solution.

1. By Fermat's little law, we have $(x+1)^n \equiv (x+1) \pmod{n}$ iff n is a prime. Also $x \equiv x^n \pmod{n}$ iff n is a prime. Combine them together we have $(x+1)^n \equiv x^n + 1$ (mid n) iff n is a prime.

Problem 7. (Chernoff Bound) Let $X_1, ..., X_t$ be independent $[0, 1]$-valued random variables, and $X = \sum_{i=1}^{t} X_i$.

1. Show that for every $r \in [0, 1/2]$, $E[e^{rX}] \leq e^{rE[X]+r^2t}$.
2. Deduce the Chernoff Bound of Theorem 2.21: Pr$[X \geq$E$[X]+\epsilon t] \leq e^{\epsilon^2 t/4}$ and Pr$[X \leq E[X] - \epsilon t] \leq e^{-\epsilon^2 t/4}$.
3. Where did you use the independence of the X_is?

Solution.

1.
2. First, observe the moment generating function of X.

$$E[e^{sX}] = \int_0^t e^{sx} f(x)dx \geq \int_a^t e^{sx} f(x)dx \geq e^{sa} P[X \geq a]$$

$$P[X \geq a] \leq \frac{E[e^{sX}]}{e^{sa}}$$

, where f is the pdf of X. Now take $a = E[X] + \epsilon t$, we get

$$P[X \geq E[X] + \epsilon t] \leq \frac{E[e^{sX}]}{e^{s(E[X]+\epsilon t)}}$$

Take $s = \frac{\epsilon}{2}$, by Item 1, we get

$$P[X \geq E[X] + \epsilon t] \leq \frac{e^{\frac{\epsilon}{2}E[X]+\frac{\epsilon^2}{4}t}}{e^{\frac{\epsilon}{2}(E[X]+\epsilon t)}} = e^{\frac{\epsilon}{2}(\frac{\epsilon t}{2} - \epsilon t)} = e^{-\frac{\epsilon^2 t}{4}} \tag{1}$$

On the other direction,

$$E[e^{-sX}] = \int_0^t e^{-sx} f(x)dx \geq \int_0^a e^{-sx} f(x)dx \geq e^{-sa} P[X \leq a]$$

$$P[X \leq a] \leq \frac{E[e^{-sX}]}{e^{-sa}}$$

Take $a = E[X] - \epsilon t$ and $s = \frac{\epsilon}{2}$, by Item 1, we get

$$P[X \leq E[X] - \epsilon t] \leq \frac{E[e^{\frac{\epsilon}{2}X}]}{e^{\frac{\epsilon}{2}(E[X]-\epsilon t)}} \leq \frac{e^{-\frac{\epsilon}{2}E[X]-\frac{\epsilon^2}{4}t}}{e^{-\frac{\epsilon}{2}(E[X]-\epsilon t)}} \leq e^{-\frac{\epsilon^2}{4}} \tag{2}$$

3. In Item 1, I use the independence of X_is to divide $E[e^{rX}]$ into $\prod_{i=1} tE[e^{rX_i}]$. Give an upper bound for each $E[e^{rX_i}]$ respectively, then integrate and get the final upper bound for $E[e^{rX}]$.