CS 221 Computational Complexity, Lecture 8 Feb 15, 2018

Randomness, Promise Problems, Randomized Complexity Classes

Instructor: Madhu Sudan *Scribe: Shyam Narayanan*

1 Topic Overview

Today, we will be talking about Randomized Computing and time complexity. We'll go over some interesting problems with randomized algorithms. We'll also go over the complexity classes ZPP, RP, coRP, and BPP and some basic properties of these classes.

2 Some Interesting Problems

A big problem that motivates randomized algorithms is that of **Primality testing**: Given $0 \leq p \leq 2^n$, determine if p is prime in poly(n) time. Algorithms such as Miller-Rabin and Solovay-Strassen are randomized algorithms that can solve this in polynomial time, though more recently Agrawal-Kayal-Saxena found a deterministic solution to primality testing.
Remark When we say randomized, the goal is for any input to output the correct answer with a high probability, not to output the correct answer for most inputs.

We now present four interesting problems:

1. Find an n-bit prime $2^{n-1} \leq p < 2^n$ in poly(n) time.

   This problem is unsolved for deterministic solutions, but has a randomized solution. By the Prime Number Thorem, picking a random element between 2^{n-1} and 2^n is prime with probability $\Theta(1/n)$, so if we try, $O(n)$ random numbers between 2^{n-1} and 2^n and run AKS on each of them (or run Miller-Rabin sufficiently many times on each of them), with high probability we will find a prime.

2. Given a prime p and an integer a, find some b such that $b^2 \equiv a \bmod p$.

   There are some randomized algorithms due to Berlekamp '72, Adleman-Manders-Miller '76, and Rabin '80, but we won't go over them. It is known that if we let p be any possibly composite number n, the problem is at least as difficult as factoring.

3. Given $k+1$ $n \times n$ matrices over the integers, $M_0, ..., M_k$, find some $r_1, ..., r_k$ such that $\det(M_0 + r_1 M_1 + ... + r_k M_k) \neq 0$.

   This problem is also quite difficult deterministically but not brobabilistically. The idea is to pick $r_1, ..., r_k$ randomly and independently from $\{1, ..., 2n\}$. The reason why this works is due to the **Schwartz-Zippel lemma** (also known as **DeMillo-Lipton lemma**) - though likely this lemma dates back much earlier, perhaps even to the 1600's. The lemma is as follows:

   Lemma 1. *If $p(x_1, ..., x_n) \in \mathbb{F}[x_1, ..., x_n]$ is a nonzero polynomial of degree at most d in any field $\mathbb{F}$, then for any finite set $S \subset \mathbb{F}$, if we pick $a_1, ..., a_n$ uniformly and independently at random from S, then $Pr[p(a_1, ..., a_n) = 0] \leq \frac{d}{|S|}$.*

   Exercise 2. *Prove the lemma!*

   To see why this is useful, note that if $S = \{1, ..., 2n\}$, then since $\det(M_0 + r_1 M_1 + ... + r_k M_k)$ has degree at most n over $r_1, ..., r_k$, choosing $r_1, ..., r_n$ randomly from S will get some $\det(M_0 + r_1 M_1 + ... + r_k M_k) \neq 0$ with probability at least $\frac{1}{2}$, assuming that $\det(M_0 + r_1 M_1 + ... + r_k M_k)$ isn't 0 uniformly.

4. **Algebraic Circuit Identity Testing (ACIT)** - given an arithmetic circuit C over $\mathbb{Z}$, does there exist $x_1, ..., x_n$ such that $C(x_1, ..., x_n) \neq 0$?

   If we were to replace C over $\mathbb{Z}$ to C over the booleans, this problem is NP-complete. However, in $\mathbb{Z}$, the problem is easier, with a randomized algorithm as follows. First, note that we can't use Schwartz-Zippel lemma directly, since a size n circuit could have degree 2^n as seen by the following diagram.

$$\begin{tikzcd} x_1 \arrow[r, bend left] \arrow[r, bend right] & \times \arrow[r] & x_1^2 \arrow[r, bend left] \arrow[r, bend right] & \times \arrow[r] & x_1^4 \arrow[r, bend left] \arrow[r, bend right] & \cdots \arrow[r] & x_1^{2^n} \end{tikzcd}$$

   This circuit is of size n but the last element has $O(2^n)$ digits and takes exponential time to compute. Therefore, we can't just try a random set $x_1, ..., x_n$ and check if $C(x_1, ..., x_n) \neq 0$. To solve this issue, we choose some prime p with $O(n^2)$ bits. Computing $C(a_1, ..., a_n)$ mod p can now be done in polynomial time, since all operations take poly(n) time. However, something with $O(2^n)$ digits can be divisible by at most $O(\frac{2^n}{n^2})$ different primes that are at least 2^{n^2} but as there are about $O(2^{n^2}/n^2)$ primes with n^2 digits, if we pick a random prime, the probability of $C(a_1, ..., a_n)$ being divisible by that prime p but not being 0 is very low. Therefore, this algorithm succeeds with high probability.

   The ACIT problem seems to be fundamental in the theory of randomized polynomial time. By that we mean it appears to be one of the hardest problems that can be solved with a randomized algorithm.

3 Randomized Complexity Classes

For understanding randomized complexity classes, we will be dealing with a lot of decision problems. Note that problems 1, 2, and 3 above are not decision problems but it is still valuable to think of many problems as decision problems for the purpose of categorizing into complexity classes.

We describe 4 complexity classes: **RP** (stands for Randomized Complexity), **CoRP** (complement of RP), **BPP** (stands for Bounded-error Probabilistic Polynomial time), and **ZPP** (stands for Zero-error Probabilistic Polynomial time).

Due to their similarity, we can define all four very similarly.

Definition 3. *For a language L, L is in the class RP if there exists a polynomial-time in expectation algorithm $M(\cdot, \cdot)$ such that $x \in L \Rightarrow Pr_y[M(x, y) = 1] \geq \frac{2}{3}$ and $x \notin L \Rightarrow Pr_y[M(x, y) = 1] \leq 0$. The part about $M(x, y) = 1$ with high probability if $x \in L$ is called completeness and the part about $M(x, y) = 1$ with low probability if $x \notin L$ is called soundness.*

L is in the class CoRP if there exists a polynomial-time in expectation algorithm $M(\cdot, \cdot)$ such that $x \in L \Rightarrow Pr_y[M(x, y) = 1] \geq 1$ and $x \notin L \Rightarrow Pr_y[M(x, y) = 1] \leq \frac{1}{3}$. It is clear to see that CoRP is the set of languges that are the complement of a language in RP.

L is in the class BPP if there exists a polynomial-time in expectation algorithm $M(\cdot, \cdot)$ such that $x \in L \Rightarrow Pr_y[M(x, y) = 1] \geq \frac{2}{3}$ and $x \notin L \Rightarrow Pr_y[M(x, y) = 1] \leq \frac{1}{3}$.

Finally, L is in the class ZPP if there exists a polynomial-time in expectation algorithm $M(\cdot, \cdot)$ such that $x \in L \Rightarrow Pr_y[M(x, y) = 1] \geq 1$ and $x \notin L \Rightarrow Pr_y[M(x, y) = 1] \leq 0$. Note that ZPP isn't quite the same as P (at least not that we know!) since these algorithms just have to be polynomial time in expectation.

We can define the classes also using the following table.

$\mathcal{C}$	RP	CoRP	ZPP	BPP
Completeness: $x \in L \Rightarrow Pr_y[M(x, y) = 1]$	$\geq \frac{2}{3}$	≥ 1	$\geq \frac{2}{3}$	≥ 1
Soundness: $x \in L \Rightarrow Pr_y[M(x, y) = 1]$	≤ 0	$\leq \frac{1}{3}$	$\leq \frac{1}{3}$	≤ 0

4 Promise Problems

We dealt with promise problems a bit in the first pset, problem 5. Recall that a promise problem is a problem of the form (Π_{YES}, Π_{NO}) with $\Pi_{YES}, \Pi_{NO} \in \{0, 1\}^*$ and $\Pi_{YES} \cap \Pi_{NO} = \emptyset$. Like how we had the complexity class RP, we can similarly define the class Promise RP as follows. If $x \in \Pi_{YES}$, the $Pr[M(x,y)=1] \geq \frac{2}{3}$, and if $x \in \Pi_{NO}$, then $Pr[M(x,y)=1] \leq 0$. There are no guarantees if $x \in \overline{\Pi_{YES}} \cap \overline{\Pi_{NO}} = \overline{\Pi_{YES} \cup \Pi_{NO}}$.

The following problem is a Promise RP-Complete problem, called Randomized Circuit SAT or RandCktSAT: Given a circuit $C : \{0,1\}^n \to \{0,1\}$, let Π_{YES} be the circuits C such that $Pr_y[C(y)=1] \geq \frac{2}{3}$ and let Π_{NO} be the circuits C such that $Pr_y[C(y)=1] \leq 0$.

Exercise 4. *Check that RandCktSAT is in Promise RP, and show that if RandCktSAT $\in P$, then $RP = P$.*

Let's go back to Problem 2 among our first 4 problems. We ask the following decision problem: Given P, a, L, U, where P is a prime and a, L, U are integers (that can be thought of mod P), does there exist b such that $L \leq b \leq U$ and $b^2 \equiv a \bmod P$? It turns out the problem is in ZPP.

Exercise 5. *Try proving the previous problem is in ZPP. As hints, use the facts that there is a probabilistic $2^{poly(n)}$ time algorithm that can output b such that $b^2 \equiv a \bmod p$, and that for any a, there are at most two congruency classes $b_1, b_2 \bmod p$ such that $b_i^2 \equiv a \bmod p$.*

5 Results about the randomized classes

The following hierarchy is quite straightforward from the definitions of the classes. Arrows indicate inclusion.

$$\begin{tikzcd}
 & & & NP \\
 & & RP \arrow[ur] \arrow[dr] & \\
P \arrow[r] & ZPP \arrow[ur] \arrow[dr] & & BPP \\
 & & CoRP \arrow[ur] \arrow[dr] & \\
 & & & CoNP
\end{tikzcd}$$

Is there anything nontrivial we can say about the hierarchy? In fact, we can! We'll show the following:

Lemma 6. *$ZPP = RP \cap CoRP$.*

Proof. Since $ZPP \subset RP$ and $ZPP \subset CoRP$, the inclusion $ZPP \subset RP \cap CoRP$ is immediate. Therefore, we just have to show any algorithm in $CoRP$ and RP is also in ZPP.

To show this, for a language $L \in RP \cap CoRP$, we can run the following program which proves it is in ZPP. Since $L \in RP$, there is some algorithm $A(\cdot,\cdot)$ such that for $x \in L$, $A(x,y)=1$ with probability at least $\frac{2}{3}$ over random y and if $x \notin L$, $A(x,y)=0$ for all y. Since $L \in CoRP$, there is some algorithm $A'(\cdot,\cdot)$ such that for $x \in L$, $A'(x,y)=1$ for all y and if $x \notin L$, $A(x,y)=0$ with probability at least $\frac{2}{3}$ over random y. Therefore, run A and A' simultaneously with a random y. Repeat until $A(x,y)=A'(x,y)$. If $x \in L$ $A(x,y)=A'(x,y)=1$ with probability $\frac{2}{3}$ and $A(x,y)=A'(x,y)=0$ will never happen. Therefore, we will always hit $A(x,y)=A'(x,y)=1$ and return 1. If $Pr_y[A(x,y)=1]=p \geq \frac{2}{3}$, then the expected number of runs needed to get $A(x,y)=A'(x,y)=1$ is $\frac{1}{p} \leq \frac{3}{2}$, so the algorithm runs in polynomial expected time. Likewise, if $x \notin L$, by symmetry we will never get $A(x,y)=A'(x,y)=1$ but will get $A(x,y)=A'(x,y)=0$ after an expected at most $\frac{3}{2}$ turns. Therefore, $L \in ZPP$, so we are done. $\square$

One question you may have about $RP, CoRP$, and BPP is why did we choose the constants $\frac{1}{3}$ and $\frac{2}{3}$? They seem somewhat arbitrary. It actually turns out that we can replace $\frac{2}{3}$ iwth $C(n)$ and $\frac{1}{3}$ with $S(n)$ as long as $C(n) \leq 1 - exp(-n^t)$ and $S(n) \geq exp(-n^t)$ for some constant t or if $C(n) = S(n) + \frac{1}{n^d}$ for some constant d. Let's prove this first for RP.

Definition 7. *Define RP_C as the class RP but with $Pr_y[M(x,y)=1]=C$. Similarly, define $CoRP_S$ and $BPP_{C,S}$.*

Theorem 8. $RP_{1/n^d} = RP_{1-exp(-n^t)}$.

Proof. The inclusion $RP_{1/n^d} \supseteq RP_{1-exp(-n^t)}$ is obvious so let's try to prove $RP_{1/n^d} \subseteq RP_{1-exp(-n^t)}$. Suppose that $M(\cdot,\cdot)$ is some algorithm that places L in RP_{1/n^d}. Then, for any $x \in L$, $Pr_y[M(x,y)=1] \le n^{-d}$. Now, pick some $y_1,..,y_k$ independently. Run the algorithm $M'(x,y)$ where $y=(y_1,...,y_k)$ that returns 1 if at least one $M(x,y_i)$ is nonzero and 0 otherwise. In this case, given $x \notin L$ it is obvious M' will output 0. If $x \in L$, the probability of us outputting 0 is at most $(1-n^{-t})^k$ as we need to fail every time. However, $(1-n^{-t})^k \le e^{-k \cdot n^{-t}}$, so if $k \ge n^{d+t}$, then we will fail with probability at most e^{-n^d}, so we are done. □

Now, we prove a slightly harder theorem, with BPP instead of RP. We will need the following important result:

Theorem 9. *(Chernoff Bound): If $Z_1,...,Z_k$ are independent, identically distributed random variables and $Z_i \in [0,1]$ such that $\mathbb{E}[Z_i]=\mu$, then*

$$Pr\left[\left|\left(\sum Z_i\right) - \mu k\right| > \lambda\sqrt{k}\right] \le e^{-\lambda^2/2}.$$

Let's see how to use the Chernoff Bound to prove the following result:

Theorem 10. $BPP_{C,C-\frac{1}{n^d}} = BPP_{1-exp(-n^t),exp(-n^t)}$.

Proof. Again, $BPP_{C,C-\frac{1}{n^d}} \supseteq BPP_{1-exp(-n^t),exp(-n^t)}$ is obvious, so we just have to show $BPP_{C,C-\frac{1}{n^d}} \subseteq BPP_{1-exp(-n^t),exp(-n^t)}$. Let Z_i be the value of $M(x,y_i)$ and let $\mu = \mathbb{E}_y[M(x,y)]$. Our algorithm for $M'(x,y)$ where $y=(y_1,...,y_k)$ that returns 1 if $\sum Z_i > k \cdot \frac{C+S}{2}$ is nonzero and 0 otherwise, where $S = C - \frac{1}{n^d}$.

To see why, this works, if $x \in L$, then $\mu \ge C$, so

$$\mathbb{P}\left(\sum Z_i < k \cdot \frac{C+S}{2}\right) \le \left(\left|\left(\sum Z_i\right) - Ck\right| \ge \frac{C-S}{2}k\right) \le e^{-O((C-S)^2 \cdot k)}.$$

We want $k \cdot (C-S)^2 \gtrsim n^t$, so we set $k \gtrsim n^{t+2d}$.

The opposite direction, for when $x \notin L$, is almost identical due to symmetry. □

6 Next Time

Next time, we'll prove a few things:

1. We'll prove that $BPP \subset P/poly$
2. We'll prove that $Promise-BPP$ equals $(Promise-RP)^{Promise-RP}$. As $Promise-RP \subset NP$, this means $BPP \subset NP^{NP} = \Sigma_p^2 \subset PH$.