6.841/18.405J Advanced Complexity Theory	February 26, 2003

Lecture 6: Randomized Algorithms, Properties of **BPP**

Instructor: Prof. Madhu Sudan — *Scribe: Shien Jin Ong*

Recap: In the previous lecture, we defined $\mathbf{ZPP}, \mathbf{RP}, \textbf{co-RP}, \mathbf{BPP}$. The following relationships between complexity classes are known.

1. $\mathbf{P} \subseteq \mathbf{ZPP} \subseteq \mathbf{RP} \subseteq \mathbf{NP}$
2. $\mathbf{P} \subseteq \mathbf{ZPP} \subseteq \textbf{co-RP} \subseteq \textbf{co-NP}$.
3. $\mathbf{RP} \cup \textbf{co-RP} \subseteq \mathbf{BPP}$.
4. $\mathbf{ZPP} = \mathbf{RP} \cap \textbf{co-RP}$.

The relationship between **BPP** and **NP** is still unknown. We, however, can prove that $\mathbf{P} = \mathbf{BPP}$ under some reasonable assumptions. Therefore, our belief is that $\mathbf{P} = \mathbf{BPP} \subseteq \mathbf{NP}$.

1 Examples of Randomized Algorithms

We give randomized algorithms for the following problems.

1. Polynomial Identity Testing.
2. Undirected Path.

1.1 Polynomial Identity Testing

We study the polynomial identity testing in the oracle model. That is given two multivariate polynomial $p(x_1, \ldots, x_n)$ and $q(x_1, \ldots, x_n)$ over a field $\mathbb{F}$, can we determine that $p = q$? This problem is equivalent to determining whether $h \stackrel{\text{def}}{=} p - q = 0$. We assume that the polynomial h is not given to us explicitly, but as an oracle O_h (black box). Given inputs $\alpha_1, \ldots, \alpha_n \in \mathbb{F}$, oracle O_h will output the value of $h(\alpha_1, \ldots, \alpha_n)$.

Next, we define the *total degree* of a polynomial. Let p be a polynomial such that

$$p(x_1, \ldots, x_n) = \sum c_{i_1 \cdots i_n} x_1^{i_1} \cdots x_n^{i_n}.$$

Then,

$$\text{total degree of } p = \max_{c_{i_1 \cdots i_n} \neq 0} \{i_1 + \cdots + i_n\}.$$

Problem (Polynomial Identity Testing): Given oracle O_h which computes the polynomial h of total degree d in n variables over a finite field $\mathbb{F}$. Does there exist $\alpha_1, \ldots, \alpha_n \in \mathbb{F}$ such that $h(\alpha_1, \ldots, \alpha_n) \neq 0$, *i.e.,* is $h(x_1, \ldots, x_n) \neq 0$?

Trivially, polynomial identity testing (PIT) can be done in $\mathbf{NP}^A$ (nondeterministic polynomial time in n, d, and $|F|$). However, PIT is not in $\mathbf{P}^A$ (exercise). We show that PIT is in $\mathbf{RP}^A$. Our randomized algorithm takes a sufficiently large subset, S, of $\mathbb{F}$ and then choose $\alpha_1, \ldots, \alpha_n$ uniformly at random from S and test whether $h(\alpha_1, \ldots, \alpha_n) = 0$.

Lemma 1 *If $p(x_1, \ldots, x_n) \neq 0$ is a polynomial of total degree d over a field $\mathbb{F}$ and $S \subseteq \mathbb{F}$, then*

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

The proof of Lemma 1 is left as an exercise. If we choose a set $S \subseteq \mathbb{F}$ such that $|S| = 2d$, our algorithm makes an error on instances $h(x_1, \ldots, x_n) \neq 0$ with probability at most 1/2. When $h(x_1, \ldots, x_n) = 0$, our algorithm never errs.

Let us do an example to illustrate the appplicability of Lemma 1.

Problem 1: What is the probability (over $x_1, \ldots, x_n$) that $x_1 \oplus x_3 \oplus x_n = 0$?

Answer: Since the operation $\oplus$ is just addition modulo 2, the probability is at most 1/2 (in fact, exactly 1/2).

Problem 2: Suppose we are given a $n \times n$ matrix M whose entries are linear equations of $x_1, \ldots, x_k$. Can we decide whether $\det(M) \equiv 0$?

Answer: Since $\det(M)$ can be evaluate efficiently when we plug in values for $x_1, \ldots, x_k$, this problem is in **RP**.

1.2 Undirected Path

Analogous with the randomized time complexity classes, we have the following randomized log-space complexity classes – **ZPL**, **RP**, **co-RL**, **BPL**. There are two major differences between a randomized log-space machine and a randomized polynomial-time machine.

1. For randomized log-space computations, we require that the machine has only one-way access to the random tape. This means that the log-space machine cannot see the previous random bits unless it has stored the random bits on its work tape tape.
2. The randomized log-space machine must halt in polynomial-time. If this requirement were to be waived, directed path can be solved in randomized log-space.

Define the undirected path problem as follows.

$$\textsc{UndirectedPath} = \{\langle G, s, t\rangle : G \text{ is an undirected graph and there exists a path from } s \text{ to } t.\}$$

Is $\textsc{UndirectedPath} \in \mathbf{L}$? While this problem is still open, we know that $\textsc{UndirectedPath} \in \mathbf{NL} \subseteq \mathbf{L}^2$. Aleliunas, Karp, Lipton, Lovasz, and Rackoff showed that $\textsc{UndirectedPath} \in \mathbf{RL}$. The randomized logspace algorithm is just the algorithm which does a random walk on the graph G.

Randomized Logspace Algorithm for UndirectedPath
On input (G, s, t), do the following.

1. `current` $\leftarrow s$.
2. for $i = 1$ to $O(V(G)^3)$
   (a) Pick a random neighbor v of the current vertex and set `current` $\leftarrow v$.
   (b) If `current` $= t$, halt and *accept*.
3. If we have not reached t after $O(V(G)^3)$ steps, *reject*.

The correctness of the above algorithm is based on the following lemma.

Lemma 2 *Let G be a connected, undirected graph on $n = V(G)$ vertices. Then, we have that*

$$\Pr[\textit{walk of length } O(n^3) \textit{ does not visit all vertices}] \leq \frac{1}{2}$$

Considering that $\textsc{UndirectedPath} \in \mathbf{RL}$, can UndirectedPath be solved in less than $(\log n)^2$ space? Thus far, we know that $\textsc{UndirectedPath} \in \mathbf{L}^{4/3}$ and $\mathbf{RL} \subseteq \mathbf{L}^{3/2}$.

2 BPP has polynomial-sized circuits

Recall that **P/poly** is the class of languages decidable by polynomial-sized circuits. Previously, we defined the class **BPP** as follows.

A language $L \in \mathbf{BPP}$ if there exist a probabilistic polynomial-time algorithm M such that

$$\begin{aligned} x \in L &\implies \Pr_r[M(x,r)=1] \geq 2/3. \\ x \notin L &\implies \Pr_r[M(x,r)=1] \leq 1/3. \end{aligned}$$

The error bound in such **BPP**-algorithm is 1/3. To prove that $\mathbf{BPP} \subset \mathbf{P/poly}$, we need to amplify the confidence (make the error bound exponentially small).

We achieve this by repeating our **BPP**-algorithm $\mathrm{poly}(|x|)$ times and taking majority vote. Using the Chernoff bound analysis, our error is reduced to $2^{-2|x|}$. Hence, an alternative formulation of **BPP** follows.

A language $L \in \mathbf{BPP}$ if there exist a probabilistic polynomial-time algorithm M such that

$$\begin{aligned} x \in L &\implies \Pr_r[M(x,r)=1] \geq 1-2^{-2|x|}. \\ x \notin L &\implies \Pr_r[M(x,r)=1] \leq 2^{-2|x|}. \end{aligned}$$

Theorem 3 *(Adelman)* $\mathbf{BPP} \subset \mathbf{P/poly}$.

Proof: Fix a language $L \in \mathbf{BPP}$ and let M be a **BPP**-algorithm for L with error bound $2^{-2|x|}$. Let χ_L be the characteristic function for L, *i.e.,* $\chi_L(x) = 1$ if $x \in L$, and $\chi_L(x) = 0$ if $x \notin L$. For each x of length n, define r to be *bad* for x if $M(x,r) \neq \chi_L(x)$.

We know that for any $x \in \{0,1\}^n$,

$$\Pr_r[r \text{ is bad for } x] \leq 2^{-2n}.$$

By the union bound, we have

$$\Pr_r[r \text{ is bad for any } x \in \{0,1\}^n] \leq 2^n 2^{-2n} = 2^{-n}.$$

Hence, there exists an r^* that is good for all $x \in \{0,1\}^n$. This means that $M(x,r^*) = \chi_L(x)$ for all $x \in \{0,1\}^n$. In addition, $|r^*| = \mathrm{poly}(n)$. The string r^* will be the nonuniform advice for deciding the language L. This shows that $L \in \mathbf{P/poly}$. $\square$