E0 224 Computational Complexity Theory **Indian Institute of Science**
Fall 2014 **Department of Computer Science and Automation**

Lecture 18: Oct 15, 2014

Lecturer: Chandan Saha <chandan@csa.iisc.ernet.in> *Scribe: Bibaswan Kumar Chatterjee*

In this lecture we look at the class BPP and co-BPP, usefulness of randomness in computation and one-sided error randomized algorithms which are captured by classes RP and co-RP which are subsets of BPP.

18.1 Class BPP

Definition 18.1. *For $T : \mathbb{N} \rightarrow \mathbb{N}$ and $L \subseteq \{0,1\}^*$ we say that a PTM M decides L in time $T(n)$ if for every $x \in \{0,1\}^*$, M halts in $T(|x|)$ regardless of its random choices and $Pr[M(x) = L(x)] \geq \frac{2}{3}$.*
*We define **BPTIME**$(T(n))$ as the class of languages decided by PTMs in $O(T(n))$ time and define **BPP** $= \bigcup_c$ **BPTIME**(n^c).*

Definition 18.2. ***(Alternative Definiton of BPP) :*** *A language $L \subseteq \{0,1\}^*$ is in **BPP** if there is a deterministic poly-time TM M and a polynomial function $q(.)$ $Pr_{r \in_R \{0,1\}^{q(|x|)}}[M(x,r) = L(x)] \geq \frac{2}{3}$*

It is easy to see the above two definitions are equivalent. Suppose a language L is in **BPP** by the first definition, it is easy to construct a TM $\overline{M}$ that given a random string r of length $\leq q(|x|)$, $\overline{M}$ will simply simulate $M(x)$ using the random bits in r and output $M(x)$. Since L is in BPP then with probability $\geq \frac{2}{3}$ we can get r such that simulation of $M(x)$ using r outputs $L(x)$. Similarly, if L is in BPP by the second definition, then $\exists$ polynomial size random string r s.t. $\overline{M}(x,r) = L(x)$ with probability $\geq \frac{2}{3}$. We can think of a PTM M which randomly generates string r and then runs $\overline{M}(x,r)$. Clearly, $Pr[M(x) = L(x)] = Pr[\overline{M}(x,r) = L(x)] \geq \frac{2}{3}$.

Conjecture 18.3. ***BPP = P***

Claim 18.4. ***BPP*** $\subseteq$ ***EXP***
*This is clear from the alternate definition of **BPP** because if we are allowed $2^{poly(n)}$ time, we can simply enumerate all possible random choices of a poly-time PTM.*

Researchers currently know that BPP is sandwiched between P and EXP but are unable to show that BPP is a proper subset of NEXP.

Question: How does NP relate to BPP?
The relation between BPP and NP is unknown. It is not known if BPP $\subseteq$ NP or NP $\subseteq$ BPP or neither. It is however known that BPP $\subseteq P_{/poly}$[Adl78], which implies (by Karp-Lipton Theorem), if **NP** $\subseteq$ **BPP**, then **PH** collapses. Another important result discovered is BPP $\subseteq \Sigma_2^p \cap \Pi_2^p$[Sip83][Laut83].

Definition 18.5. ***co-BPP:*** *A language $L \subseteq \{0,1\}^*$ is in **co-BPP** iff $\overline{L} \in$ **BPP**.*

Lemma 18.6. ***BPP*** $=$ ***co-BPP***.

Proof. Let $L \subseteq \{0,1\}^*$ be a language such that $L \in co - BPP$. Then by the definition of co-BPP,

$$\begin{aligned} L \in co - BPP &\Leftrightarrow \overline{L} \in BPP \\ &\Leftrightarrow \exists poly - time\ PTM\ M\ s.t.\ Pr[M(x) = \overline{L}(x)] \geq \frac{2}{3} \\ &\Leftrightarrow \exists poly - time\ PTM\ \overline{M}\ s.t.\ \overline{M}(x) = \neg M(x)\ \ \&\ Pr[\overline{M}(x) = L(x)] \geq \frac{2}{3} \\ &\Leftrightarrow \exists poly - time\ PTM\ \overline{M}\ s.t.\ Pr[\overline{M}(x) = L(x)] \geq \frac{2}{3} \\ &\Leftrightarrow L \in BPP \end{aligned}$$

□

Thus we see the class BPP is closed under complement.

18.2 Usefulness of Randomness in computation

In this section we will explore how randomization can lead to simple algorithms for problems with very efficient run-time complexity.

Definition 18.7. ***Expected running time:*** *Let M be a P.T.M. that decides a language $L \subseteq \{0,1\}^*$. For a string $x \in \{0,1\}^*$ let T_x be the time taken by M to decide if $x \in L$ where T_x is a random variable. We say the* ***expected running-time*** *of M is $T(n)$ iff $\mathbb{E}[T_x] \leq T(|x|)$ for every $x \in \{0,1\}^*$.*

18.2.1 Finding the k^{th} smallest element in an unsorted array

It is known that selecting the k^{th} smallest element in an unsorted array can be done in $O(n)$ time using the linear time selection algorithm[CLRS01]. But its analysis is quite complicated and it is also quite difficult to implement in practice. Here we give a simple linear time randomized algorithm for selecting the k^{th} smallest element in an unsorted array which is much simple to implement as well as analyze.

Find(A,k)

Input: $A = \{a_1, \ldots, a_n\} \in \mathbb{Z}^n$, $k \in \mathbb{Z}^+$ and $k < n$
Output: the k^{th} smallest element in A

1: Pick i uniformly at random from $[n]$
2: **DIVIDE** A into three parts: $A_1 = \{a_j \in A/\{a_i\}$ AND $a_j \leq a_i\}$, $A_2 = \{a_j \in A/\{a_i\}$ AND $a_j > a_i\}$ and element a_i. Let $m = |A_1|$.
3: **If** m is $k-1$ then **output** a_i
3: **Else If** $m \geq k$ then
4: **Call** Find(A_1, k)
5: **Else**
6: **Call** Find($A_2, k - m$)

Theorem 18.8. *The expected running-time of Find(A, k) is $O(n)$ where $n = |A|$.*

Proof. Let $T(n)$ be the running time of Find(A, k).
Let us define an indicator variable,

$$\begin{aligned} I_j &= 1 \; if \; j = m \; (m \; defined \; in \; our \; algorithm) \\ &= 0 \; otherwise \end{aligned}$$

So, $\forall j \in [n]$

$$\begin{aligned} \mathbb{E}\left[I_j\right] &= Pr(m = j) \\ &= \frac{1}{n} \end{aligned}$$

Then,

$$T(n) \leq cn + \sum_{j \geq k} [I_j \times T(j)] + \sum_{j < k-1} [I_j \times T(n-j)]$$

where c is a constant.

$$\begin{aligned} \mathbb{E}[T(n)] &\leq cn + \sum_{j \geq k} \left[\mathbb{E}\left[I_j\right] \times \mathbb{E}\left[T(j)\right]\right] + \sum_{j < k-1} \left[\mathbb{E}\left[I_j\right] \times \mathbb{E}\left[T(n-j)\right]\right] \\ \mathbb{E}[T(n)] &\leq cn + \sum_{j \geq k} [\frac{1}{n} \mathbb{E}[T(j)]] + \sum_{j < k-1} [\frac{1}{n} \mathbb{E}[T(n-j)]] \end{aligned}$$

Now here we make an inductive assumption that our $\mathbb{E}[T(n)] = \alpha cn$ where α is some constant > 1. We can assume this is to be trivially true for $\mathbb{E}[T(1)]$. We show that if our assumption is true for $T(j)$ where $j < n$, then its true for $T(n)$. Going back to our proof,

$$\begin{aligned} \mathbb{E}[T(n)] &\leq cn + \frac{1}{n} [\sum_{j \geq k} \mathbb{E}[T(j)] + \sum_{j < k-1} \mathbb{E}[T(n-j)]] \\ \mathbb{E}[T(n)] &\leq cn + \frac{\alpha c}{n} [\sum_{j \geq k} j + \sum_{j < k-1} (n-j)] \\ \mathbb{E}[T(n)] &\leq cn + \frac{\alpha c}{n} [\frac{n(n+1)}{2} - \frac{k(k-1)}{2} + (k-1)n - \frac{n(k-1)}{2}] \\ \mathbb{E}[T(n)] &\leq cn + \frac{\alpha c}{n} [\frac{n^2 + (2k-1)n - 2k^2 - 2k}{2}] \\ \mathbb{E}[T(n)] &\leq cn + \frac{\alpha c}{n} [\frac{n^2(\alpha - 1)}{\alpha}] \; for \; some \; large \; n > n_0 \\ \mathbb{E}[T(n)] &\leq cn + (\alpha - 1)cn \\ \mathbb{E}[T(n)] &\leq \alpha cn \\ \mathbb{E}[T(n)] &= O(n) \end{aligned}$$

□

Remark: Whether or not random bits are absolutely indispensable depends on the context. For cases like

- **Interactive Protocols**
- **Probabilistically checkable Proofs**

random bits are absolutely essential. Next we will explore an example from coding theory where randomization is very useful.

18.2.2 Hadamard Codes

Hadamard Codes is an error-correcting code that is used for error detection and correction when transmitting messages over noisy or unreliable channels.

HadamardCode(x)
Input: $x \in \{0,1\}^k$
Output: $y \in \{0,1\}^{2^k}$
for $i = 0, 1, \ldots, 2^k - 1$
$y_i = \bigoplus_{\lfloor i \rfloor_j = 1} x_j$
end

Now let us look at the problem of recovering each bit x_i of x given a possibly corrupted version of $y = HadamardCode(x)$ since we are looking at the context of transmission in noisy channel. Let us consider that ρ fraction of the bits in y are corrupted. Our task is to recover each bit x_i making as few reads of the bits of y as possible.

Decode(y,i)
Input: $y \in \{0,1\}^{2^k}$ where y is $HadamardCode(x)$ with atmost ρ fraction of bits corrupted and $i \in [k]$
Output: $\overline{x}_i$ such that $Pr[\overline{x}_i = x_i]$ with high probability
1: Pick a random subset s of $[n]$
2: Read y_s from y
3: Read $y_{s\triangle\{i\}}$ from y
4: Output $\overline{x}_i = y_s \oplus y_{s\triangle\{i\}}$

Analysis:
If y_s and $y_{s\triangle\{i\}}$ are not corrupted then,

$$y_s = \bigoplus_{j \in s} x_j$$

$$y_{s\triangle\{i\}} = \bigoplus_{j \in s\triangle\{i\}} x_j$$

$$y_s \oplus y_{s\triangle\{i\}} = x_i$$

This is the ideal case. We have assumed that possibly ρ fraction of the bits in y are corrupted.

$$Pr[y_s \text{ is corrupted }] \leq \rho$$
$$Pr[y_{s\triangle\{i\}} \text{ is corrupted }] \leq \rho$$
$$Pr[y_s \text{ or } y_{s\triangle\{i\}} \text{ is corrupted }] \leq 2\rho \text{ (by union bound)}$$

Thus, $Pr[\overline{x}_i = x_i] \geq (1-2\rho)$ which is high provided ρ is small.

18.3 Class RP and co-RP

The class BPP captures probabilistic algorithms with two sided error. A PTM M that computes a language $L \in BPP$ can output 1 when the input string does not belong to L and output 0 when the input string does belong to L with some small probability. However, there are probabilistic algorithms which have one-sided error.

Definition 18.9. *For T : $\mathbb{N} \to \mathbb{N}$ and $L \subseteq \{0,1\}^*$ we say that $L \in$ RTIME$(T(n))$ if $\exists$ a PTM M such that for every $x \in \{0,1\}^*$, M halts in $T(|x|)$ time and*

$$x \in L \Rightarrow Pr[M(x) = 1] \geq \frac{2}{3}$$
$$x \notin L \Rightarrow Pr[M(x) = 0] = 1$$

Definition 18.10. ***Class RP:*** $RP = \bigcup_c RTIME(n^c)$

Definition 18.11. ***Class co-RP:*** *A language $L \in co-RP$ iff $\overline{L} \in RP$*

It is clear form the definition that both RP and co-RP are subsets of BPP.

Recall the Polynomial Identity Testing problem in the last lecture.
PIT: $L = \{C :$ the polynomial computed by circuit C is identically $0\}$.
The algorithm which was discussed was such that for $x \in \{0,1\}^*$

$$x \in L \Rightarrow Pr[M(x) = 1] = 1$$
$$x \notin L \Rightarrow Pr[M(x) = 0] \geq \frac{2}{3}$$

This shows $PIT \in co-RP$.

References

[adl78] ADLEMAN, LEONARD, 'Two theorems on random polynomial time.' *2013 IEEE 54th Annual Symposium on Foundations of Computer Science. IEEE,*, 1978

[Sip83] SIPSER, MICHAEL, 'A complexity theoretic approach to randomness' *Proceedings of the 15th ACM Symposium on Theory of Computing (ACM Press): 330335.*, 1983

[Laut83] LAUTEMANN, CLEMENS 'BPP and the polynomial hierarchy' *Inf. Proc. Lett. 17 (4): 215217.*, 1983

[CLRS01] THOMAS H. CORMEN, CHARLES E. LEISERSON, RONALD L. RIVEST and CLIFFORD STEIN, 'Introduction to Algorithms, Second Edition.' *MIT Press and McGraw-Hill, 2001. ISBN 0-262-03293-7. Chapter 9: Medians and Order Statistics, pp.183196.* 2001