COM S 6810 Theory of Computing February 17, 2009

Lecture 9: Randomness and Computation

Instructor: Rafael Pass *Scribe: Matt Weinberg*

1 Motivation

Proposition 1 (Strong Church-Turing Hypothesis) *Every physically realizable form of computation can be simulated by a Turing Machine with polynomial-time overhead.*

Note that this not a rigorous hypothesis. For instance, "physically realizable" is not a well-defined term, and we don't even know what forms of computation are physically realizable. However, this motivates the discussion of randomized computing methods versus deterministic ones.

2 Examples of Randomized v. Deterministic

2.1 Example 1: Primality Testing

The Problem: Given an n-bit integer, N, determine in polynomial time (in n) if N is prime.

Miller Test: The Miller Test for Primality deterministically determines if N is prime in polynomial time. But the correctness of the algorithm is conditional upon the unproven Riemann Hypothesis.

Miller-Rabin Test: The Miller-Rabin test for Primality probabalistically determines if N is prime in polynomial time. This algorithm is no longer conditional upon the Riemann Hypothesis, but is also no longer deterministic.

Agrawal-Kayal-Saxena (AKS) Test: The AKS Test for Primality deterministically determines in polynomial time if N is prime. This algorithm does not depend upon the Riemann Hypothesis, and is still deterministic.

Significance: The Miller-Rabin test appeared in 1975, and in 2002 the AKS test appeared. In the end, a deterministic algorithm existed that could solve the same problem.

2.2 Example 2: Finding Primes

The Problem: Given an n-bit integer, N, find any x such that x is prime and $x \in [N+1, N^2]$.

Currently there is no known deterministic polynomial time algorithm that can solve this problem. However there are two useful theorems that make randomized algorithms a possibility.

Definition 1 *Let $\pi(x)$ denote the number of primes less than or equal to x.*

Theorem 1 (Prime Number Theorem)

$$\lim_{x \to \infty} \pi(x) = \frac{x}{\ln(x)}$$

Theorem 2 (Chebyshev's Theorem)

$$\pi(x) \geq \frac{x}{2\log(x)}$$

Using the Prime Number Theorem, we can say that there are about $\frac{N^2}{\ln(N^2)}$ primes less than or equal to N^2, and about $\frac{N}{\ln(N)}$ primes less than or equal to N, so the chance that a random number between N+1 and N^2 is prime is at least $\frac{1}{\ln(N)}$ - $\frac{1}{N\ln(N)}$, which is at least $\frac{1}{2n}$ for large N.

Instead we could just use Chebyshev's Theorem, and instead know that there are at least $\frac{N^2}{2\log(N)}$ primes less than or equal to N^2, and at most N primes less than or equal to N. So there are at least $\frac{N^2}{2\log(N)} - N$ primes between N+1 and N^2, and the chance that a random number between N+1 and N^2 is prime is at least $\frac{1}{2\log(N)}$ - $\frac{1}{N}$, which is at least $\frac{1}{4n}$ for large N.

This claim lends itself nicely to a randomized algorithm. We can just randomly sample elements of [N+1,N^2], and in expectation we only need to sample 2n times before we find one. Using the the AKS primality test, we can test each sample for primality in polynomial time as well.

Significance: Here we have the first example of a problem that can be solved efficiently in expectation by a randomized algorithm, and we do not know of a polynomial time deterministic algorithm. We do not, however, know that one does not exist.

2.3 Example 3: Quadratic Residue

The Problem: Given an n-bit prime number, p, and another number, a, such that a has a square root mod p, ie: ($\exists$ x)($x^2 \equiv$ a (mod p)), find such an x.

Again, there is no known polynomial time algorithm to solve this problem, but it is easily solved with randomization.

2.4 Example 4: Polynomial Equality

The Problem: Given two multi-variable polynomials, over a finite field, F, $P_1(x_1, x_2, .., x_n), P_2(x_1, x_2, ..$ decide if $(\forall x_1, x_2, .., x_n)(P_1(x_1, x_2, .., x_n = P_2(x_1, x_2, .., x_n))$

Note that this is not a trivial problem, the polynomials may be written using different representations, and in the example of $(x_1 + x_2 + ... + x_n)^n$, it may require exponential time just to write out a standardized representation of the polynomial. Additionally, this problem can be reduced to Polynomial Identity Testing, ie: Given $P(x_1, x_2, .., x_n)$, decide if $(\forall x_1, x_2, ..., x_n)(P(x_1, x_2, .., x_n) = 0)$ by letting $P(x_1, x_2, .., x_n) = P_1(x_1, x_2, .., x_n) - P_2(x_1, x_2, .., x_n)$. So we will look at Polynomial Identity Testing.

Consider a version of this problem where P is given in the form of a black box oracle, O, that can evaluate P on any input.

Proposition 2 *$PolyID^O \in coNP^O$. Consider the problem of proving that a polynomial is not identically zero. Then there $\exists(x_1, .., x_n)$ such that $P(x_1, .., x_n) \neq 0$. So an NTM would non-deterministically evaluate $P(x_1, .., x_n)$ for each $(x_1, .., x_n)$, and output yes iff $P(x_1, .., x_n) \neq 0$.*

Proposition 3 *$PolyID^O \in RP^O$. The algorithm simply picks random elements of F^n and evaluates P. If P is ever non-zero, we output no. If P is always 0 on every input we test, we output yes. This algorithm only works when $|F| > d$, where d is the total degree of P. The correctness of this algorithm is based on the following Lemma.*

Lemma 3 (DeMillo-Lipton-Schwartz-Zippel) *If $P(x_1, .., x_n)$ is an n-variable, non-zero polynomial of total degree d over a finite field F, then P has at most $d|F|^{n-1}$ roots.*

Corollary: If P is a non-zero polynomial, then $Pr(P(x_1, .., x_n) = 0) \leq \frac{d}{|F|}$. In particular, if $|F| > 3d$, then $Pr(P(x_1, .., x_n) = 0) \leq \frac{1}{3}$

Because of this Corollary, we can randomly sample elements of F^n and have at least a $1 - (\frac{d}{|F|})^k$ chance that of being correct.

Significance: In this case, it has actually been proved that no deterministic algorithm can solve this problem in polynomial time, so here we have the first example of a case that shows the power of randomness. However, under other reasonable forms of input, it is not known whether this problem is in P.

2.5 Example 5: Semantically Secure Encryption

This is another example where it has been proven that no deterministic method can create a Semantically Secure Encryption, but there is a known Randomized Algorithm that can.

2.6 Summary

In Summary, all of these problems can be solved using randomization because mathematics has proved a lot of theorems related to density of objects (such as primes, or zeroes of a non-zero polynomial). However, very little is known about the actual distribution of the same objects, which is what would be needed to help deterministic algorithms. So it may be the case for the examples where it hasn't been proven that no polynomial deterministic algorithm exists that a deterministic algorithm will be found when more is known about the distribution of primes and zeroes.

3 Definitions and Classes

3.1 Definitions

Definition 2 (Randomized Turing Machine-1): *A Turing Machine that also has access to an infinite tape. Each entry on the tape is a single random bit. The machine can only read in one direction.*

Definition 3 (Randomized Turing Machine-2): *A Turing Machine that also has access to random oracle. The oracle will give a single random bit on command each time it is requested by the machine.*

Proposition 4 *The two above definitions are equivalent. For each infinite tape, an oracle exists that outputs random bits in that order. And for each oracle, an infinite tape exists that records its output sequentially. The reason we dont allow the machine in definition 1 to read backwards is because this may create issues with space complexity.*

Definition 4 (Probabilistic Polynomial Time): *A randomized Turing Machine that always terminates in polynomial time.*

3.2 Classes

Definition 5 (Randomized Polynomial Time): *This is the class of languages, L, such that ∃ a PPT-RTM, M such that:*

$$(\forall x)(x \in L \rightarrow Pr[M(x) = 1] \geq \frac{1}{2})$$

$$(\forall x)(x \notin L \rightarrow Pr[M(x) = 0] = 1$$

Definition 6 (Zero Probabilistic Polynomial): *This is the class of languages, L, such that ∃ a PPT-RTM, M such that:*

$$(\forall x)(Pr[M(x) =?] \leq \frac{1}{2})$$

$$(\forall x)(\forall r)(M_r(x) \neq? \rightarrow M_r(x) = L(x))$$

ZPP is the class of languages such that there exists a RTM that is always correct, and has polynomial running time in expectation. This is a simple application of Markov's inequality, because we output ? at most once in expectation before we output 1 or 0.

Definition 7 (Bounded Probabilistic Polynomial): *This is the class of languages, L, such that ∃ a PPT-RTM, M such that:*

$$(\forall x)(x \in L \rightarrow Pr[M(x) = 1] \geq \frac{2}{3})$$

$$(\forall x)(x \notin L \rightarrow Pr[M(x) = 0] \geq \frac{2}{3})$$

3.3 Observations

Proposition 5 (RP ∈ NP): *(∀L ∈ RP)(∀x ∈ L)(∃ r)(M_r(x)=1). So consider an NTM that guesses the random bit string, r, then runs M_r. Then$(x \notin L) \rightarrow (\forall r)(M_r(x) = 0)$, so the NTM will reject all $x \notin L$. Furthermore, $x \in L \rightarrow (\exists r)(M_r(x) = 1)$, so at least one path on the NTM will accept x.*

Proposition 6 *We can replace the $\frac{1}{2}$ in the definition of RP or ZPP by $1 - \frac{1}{2^n}$ or $\frac{1}{n^c}$*
Proof. *Given a PPT-RTM for $L \in RP$, M, let M' be the following machine: Run M n times. If M(x)=0 every single time, output 0. If M(x)=1 on any trial, output 1. Now we know that because M is a PPT-RTM for $L \in RP$, if $x \notin L$, M(x)=0, always. So M' will always output 0. If $x \in L$, then the chance that M(x)=0 on every trial is $1 - \frac{1}{2^n}$. So M' has running time equal to n times that of M, so M' is still a PPT-RTM, and $x \in L \rightarrow Pr[M'(x) = 1] \geq 1 - \frac{1}{2^n}$.*
Given a language, L, such that $(\forall x)(x \in L \rightarrow Pr[M(x) = 1] \geq \frac{1}{n^c}$, let M' be the following machine: Run M n^c times. If M(x)=0 every single time, output 0. If M(x)=1 on any trial, output 1. Using the same logic as above, M' is correct and has running time equal to that of n^c times that of M.
Furthermore, $x \in L \rightarrow Pr[M(x) = 1$ at least once$] \geq 1 - (1 - \frac{1}{n^c})^{n^c} \geq \frac{1}{2}$.
The exact same proof works in both cases for ZPP. ■

Proposition 7 *coRP can be described as all languages L such that $\exists M$ such that:*

$$(\forall x)(x \in L \rightarrow Pr[M(x) = 1] = 1)$$

$$(\forall x)(x \notin L \rightarrow Pr[M(x) = 0] \geq \frac{1}{2})$$

Consider any $L \in RP$. Then consider M that decides L. Let $M'(x) = 1 - M(x)$. Then M' decides $\overline{L}$ with the required probabilities.

Proposition 8 *$RP \cap coRP = ZPP$*
Proof. *For any M deciding $L \in ZPP$, let M' be defined as follows. Run M. If M(x) = 1, $M'(x) = 1$. If M(x) = 0, $M'(x) = 0$. If $M(x) =?$, $M'(x) = 0$. Then M' has the required probabilities for $L \in RP$.*
Let M'' be defined as follows. Run M. If M(x) = 1, $M''(x) = 1$. If M(x) = 0, $M''(x) = 0$. If $M(x) =?$, $M''(x) = 1$. Then M'' has the required probabilites for $L \in coRP$.
So $\forall L \in ZPP$, $L \in RP \cap coRP$. For any $L \in RP \cap coRP$, let M and M' be the machines deciding L. Let M'' be defined as follows. Run M and M'. If M(x) = $M'(x)$ = 1, output 1. If M(x)=$M'(x)$ = 0, output 0. Else output ?. Then say $x \in L$. Then $M'(x) = 1$, so we will never output 0. Furthermore, Pr[M(x)=1] $\geq \frac{1}{2}$, and we output ? only when M(x) = 0. So Pr[$M''(x) =?$] $\leq \frac{1}{2}$. Say $x \notin L$. Then $M(x) = 0$, so we will never output 1. Furthermore, $Pr[M'(x) = 0] \geq \frac{1}{2}$. So Pr[$M''(x) =?$] $\leq \frac{1}{2}$.
So $\forall L \in RP \cap coRP, L \in ZPP$. ■

Proposition 9 *The $\frac{2}{3}$ in BPP can be replaced by $\frac{1}{2} + \frac{1}{n^c}$ or $1 - \frac{1}{2^n}$.* **Proof.** *Let M be a RTM such that $x \in L \rightarrow Pr[M(x) = 1] \geq \frac{1}{2} + \frac{1}{n^c}$. And $x \notin L \rightarrow Pr[M(x) = 0] \geq \frac{1}{2} + \frac{1}{n^c}$. Let M' be defined as follows. Run M n^{2c} times. If M output 1 on at least half of the executions, output 1, otherwise output 0. Now let X_i denote the random variable that is*

1 when the i^{th} execution of M(x) = 1, 0 when M(x) = 0. Then $x \in L \rightarrow E[X] \geq \frac{1}{2} + \frac{1}{n^c}$. So using Chernoff's bound,

$$Pr[|\sum_{i=1}^{2n^c} \frac{X_i}{2n^c} - E[X]| \geq \frac{1}{n^c}] \leq 2^{-\frac{1}{n^{2c}} * 2n^{2c}}$$

So in order for M' to be wrong, it must be the case that $\sum_{i=1}^{2n^c} \frac{X_i}{2n^c} \leq \frac{1}{2}$. ie: $|\sum_{i=1}^{2n^c} \frac{X_i}{2n^c} - E[X]| \geq \frac{1}{n^c}$, because E[X] $\geq \frac{1}{2} + \frac{1}{n^c}$. So the chance that M' is wrong is bounded by the Chernoff bound, which is 2^{-2} or $\frac{1}{4}$. So we are right with probability at least $\frac{3}{4}$. The same argument holds when M(x) = 0. This is still polynomial time, because we just simulate $M\ n^{2c}$ times.
The same argument as above holds if given an RTM, M, such that $x \in L \rightarrow Pr[M(x) = 1] \geq \frac{2}{3}$, $x \notin L \rightarrow Pr[M(x) = 0] \geq \frac{2}{3}$ and we want to show that M' which simulates $M\ 9n$ times is correct with probability at least $1 - \frac{1}{2^n}$. ■

3.4 Conjectures and Theorems

Proposition 10 *(Conjecture) BPP = P. In fact, it is even unknown if $BPP \subseteq NP$.*

Proposition 11 *$BPP \subseteq P/Poly$. Will be proved next lecture.*

Proposition 12 *$BPP \subseteq \Sigma_2 \cap \Pi_2$. Will be proved next lecture.*