**CIS 511: Theory of Computation** **Mar 28, 2017**

# Lecture 18: Randomized Complexity

*Professor Sampath Kannan* *Zach Schutzman*

**NB**: *These notes are from CIS511 at Penn. The course followed Michael Sipser's Introduction to the Theory of Computation (3ed) text.*

## Randomized Algorithms

**Definition 18.1** *A **randomized algorithm** is one in which we can toss random coins.*

*We don't worry about how this randomness is produced, we just take it as given that we can make these queries from an idealized source of randomness.*

***Definition 18.2*** *A **randomized** or **probabilistic Turing machine** may either move deterministically or reach a state where it can flip a coin and move according to the random output. A probabilistic Turing machine $M$ is said to accept a string $x \in L$ with probability equal to the sum of the probabilities of the leaves where $M$ produces the correct answer.*

***Definition 18.3*** *The class **Bounded Probabilistic Polynomial time (BPP)** is the class of languages recognizable by a randomized Turing machine which accept on inputs $x \in L$ with probability at least $\frac{2}{3}$.*

***Claim 18.4*** *We can equivalently define $BPP$ as having error $0 \leq \epsilon < \frac{1}{2}$.*

**Proof:** *Suppose we have a machine $M$ with error $\epsilon < \frac{1}{2}$. Then we show there is an $M'$ with error $\frac{1}{2^n}$ on inputs of size $n$. We simply have $M'$ run $M$ $2k$ times on input $x$ and outputs the majority answer, where $k$ is constructed as follows:*

*To see this, consider our sequence of $M$'s outputs, e.g. $cccwwcwccwc\ldots$. A sequence with more than $k$ $w$s will cause an error in $M'$. We want this to occur with probability at most $\frac{1}{2^n}$. Any particular sequence with some number of $c$s and $w$s is $\epsilon^{\#w}(1-\epsilon)^{\#c}$. So $M'$ makes a mistake when $\#w \geq k$. But this is less than $\epsilon^k(1-\epsilon)^k$. There are at most $2^{2k}$ 'bad' sequences (this is all of them!). Thus, the total probability of all bad sequences is less than or equal to $2^{2k}\epsilon^k(1-\epsilon)^k = (4\epsilon(1-\epsilon))^k$. This is strictly less than one whenever $\epsilon < \frac{1}{2}$.*

*We can then choose $k$ to make the lower bound as low as we want for an equivalent definition of $BPP$.* ■

*What is the space complexity of probabilistic Turing machines?*

*Consider the language $UPATH = \{\langle G, s, t\rangle |$ there is an $st$ path in $G\}$. This is in randomized log-space ($RL$). We can do this by randomly choosing a neighbor at each step, not exceeding $n^3$ steps on any branch.*

### Polynomial Identity

*The problem of polynomial identity testing asks whether two multivariate polynomials $f$ and $g$ on $n$ variables are exactly identical. We have two black boxes, one for $f$ and one for $g$. We want to know if $f = g$.*

*Denote $L_{non-id} = \{\langle f, g\rangle | f \neq g\}$ for polynomials $f$ and $g$. If $f$ and $g$ are univariate of degree less than or equal to $d$, we can know if $f = g$ just by testing $d + 1$ different values.*

***Lemma 18.5*** *(Schwartz-Zippel) Let $h$ be a non-zero polynomial on $n$ variables of degree $d$. If values $x_1 \ldots x_n$ are chosen uniformly and independently at random from a set $S$ of size $n$, then the probability that $h(x) = 0$ is less than or equal to $\frac{d}{n}$.*

*If we think of $h = f - g$, then we have the 'bad' event of getting unlucky of picking $d + 1$ points where $f(x) = g(x)$ even when $f \neq g$ is sufficiently small.*

*Now we can come up with a randomized algorithm to recognize $L_{not-id}$. We choose a set $S$ of $3d$ points from the domain of the variables. For each $x_i$ pick uniformly at random some $r_i \in S$. Now, evaluate $f(r)$ and $g(r)$. If we get different answers, accept. Otherwise, reject. If $f$ and $g$ are identical, then the algorithm always correctly rejects. If $f$ and $g$ are different, the probability we picked a 'bad' point and make a mistake is at most $\frac{1}{3}$, because we have $\frac{d}{3d}$ probability by the lemma.*