Recent advances in Complexity CIS 6930/CIS 4930	October 31, 2002
Lecture 20	
Lecturer: Dr. Meera Sitharam	Scribe: Hongyu Guo

1 Hardness vs. Randomness

We first define hardness. We say a function h is hard if it is hard to find a good approximation with any functions in a certain complexity class C. More precisely

$$\forall g \in C, Prob_x[g(x) = h(x)] \leq \delta,$$

where $x \in \{0,1\}^n$ and δ may depend on n, like $\frac{1}{n^k}$ or $\frac{1}{2^{\sqrt{n}}}$.

We want to use h to create a pseudo random generator G_h whose output is a set of pseudo random strings that look like random to functions in C, meaning

$$\forall g \in C, |Prob_x[g(x) = 1] - Prob_y[g(G_h(y)) = 1]| \leq \delta'$$

That is, the pseudo random generator can fool all the functions in C.

2 Derandomization

Definition 1 *Monte Carlo randomized computation M takes $x \in \{0,1\}^n$ and a random string $y \in \{0,1\}^m$ as input and outputs 0 or 1, and*
$x \in S \Rightarrow Prob_y[M(x) = 1] \geq \frac{1}{2} + \epsilon$, and
$x \notin S \Rightarrow Prob_y[M(x) = 0] \geq \frac{1}{2} + \epsilon$,
where $\epsilon > 0$ is independent of $|x|$.

Sets S, or Boolean functions $\chi_S(x) = 1$ if $s \in S$ and $\chi_S(x) = 0$ if $x \notin S$, for which such and M polynomial in $|x|$ and $|y|$ exists, constitute a complexity class BPP (Bounded error Probabilistic Polynomial). These algorithms are always fast but probably correct.

It is still not known if $NP \subseteq BPP$ or $BPP \subseteq NP$. However, it is the general belief that $BPP = P$.

Definition 2 *Las Vegas randomized computation M takes $x \in \{0,1\}^n$ and a random string $y \in \{0,1\}^m$ as input and outputs 0 or 1, and*
(1) $Prob_y[M(x) = \chi_S(x)] = 1$, and
(2) $Prob_y[M takes time \leq n^k] \geq \frac{1}{2} + \epsilon_k$.

Sets S for such an M exists constitute complexity class ZPP (Zero error Probably Polynomial). These algorithms are always correct but probably fast.

Exercise 1 *Show $ZPP \subseteq BPP$.*

We can use the pseudorandom generator for three applications:

1. Derandomization of a single randomized algorithm A

For a given single randomized algorithm A with bounded error ϵ running in $DTIME(T(n))$, if there exists a pseudo random generator $G_A : \{0,1\}^m \longrightarrow \{0,1\}^n$ such that

(c1)(pseudorandomness)

$|Prob[A(x) = 1] - Prob[A(G(y)) = 1]| \leq \delta \leq \epsilon$,

(c2) (size of seeds)

$m \leq log(T(n))$,

(c3) (efficiency)

Running time G satisfies

$2^m \cdot runtime(G) \leq T(n)$.

We can use G_A to derandomized the algorithm A. We can simply exhaust y so that the computation is deterministic.

Note without the efficiency constraint, we can always find trivial derandomizations of any algorithms.

2. Derandomization of randomized algorithms in a complexity class C

$\forall A \in C, \quad \exists G_A$ such that conditions c1, c2 and c3 hold.

For example, to show $P = BPP$, we need to show $2^m \cdot runtime(G) \leq poly_A(n)$.

Another example, to show $BPP \subseteq DTIME(n^l ogn)$, we need $T(n) = poly_A(logn)$.

One more example, to show $BPP \subseteq DTIME(2^{n^\epsilon})$, where $0 < \epsilon < 1$, we need $T(n) = 2^{n^{\epsilon_A}}$.

3. Cryptography

(c4)

$\exists G_C, \quad \forall A \in C$, condition c1 holds, and the runtime of G_C has to be polynomial in m.

Notice the condition here is stronger than that in 2, — derandomization for a complexity class. In derandomization of a complexity class, it suffices to find one (different) pseudo random generator for each algorithm A, while in cryptography application, we need one single pseudo random generator that can cheat all the algorithms in the class.

There is another version of weaker requirement: we find a one-way function G. Computing $G(y)$ is easy while computing $G^{-1}(y)$ is hard. This means that the adversory is not able to decode.

Exercise 2 *Show that if A cannot tell the difference between $G_C(y)$ and random strings, then it cannot decode $G_C(y)$. Or in other words, if c4 holds for some $G_C, \forall A \in C$, then A cannot compute G_C^{-1}.*