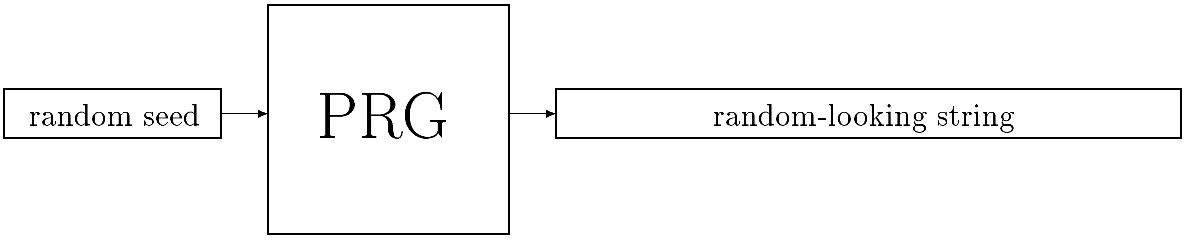
CS395T Pseudorandomness and Combinatorial Constructions
Lecture 3, January 23, 2001
Lecturer: Prof. David Zuckerman, Scribe: Vladimir Trifonov

More examples of randomized algorithms

- **Algebra:** Polynomial time factorization, square roots, primality testing, etc.
- **Approximation algorithms:** Volume determination, permanent computation, etc.
- **Graph algorithms:** Min-cut, minimum spanning tree, etc.
- **Linear programming**

Examples of randomized algorithms can be obtained from the book *Randomized Algorithms*, by R. Motwani and P. Raghavan. The important question for this course is: can randomized algorithms be derandomized efficiently?

1 Pseudorandom Generators



A pseudorandom generator(PRG) is a deterministic algorithm that converts a short random string (traditionally called *seed*) into a long string that looks random. In other words it is a function $G : \{0,1\}^s \to \{0,1\}^r$, $s \ll r$, computable by an efficient deterministic algorithm and whose output is indistinguishable from uniform by randomized algorithms. Notice that the output of the PRG will not be completely random, i.e. uniform, because the size of its range $|G(\{0,1\}^s)| \leq 2^s \ll 2^r$. A PRG can be used to decrease the number of random bits used by a randomized algorithm and, if the seed is sufficiently small, for non-trivial derandomization.

1.1 Classical Pseudorandomness

The output is obtained by concatenating strings $a_1, \ldots, a_n$, such that $a_{i+1} = f(a_i)$, for some function f (e.g. $f(x) = (rx + s) \bmod m$, linear congruential generator). The evidence that the output looks random is some ad hoc statistical test, e.g. long period. This is not a good way of proving pseudorandomness, because it is easy to come up with sequences which do not look random, but pass the test.

1.2 Modern View

According to the modern view of pseudorandomness the evidence that the output of a PRG looks random is a formal mathematical proof. For this we need a definition of what it means for the output to look random. Because we only care about fooling certain randomized algorithm $A(x, y)$, where x is the input to the algorithm and y is the random bits it uses, we will tailor this definition so that A is fooled by G, if the output behavior of A on the output of the PRG is almost the same as the output behavior of A on uniformly chosen random bits. Ideally we want to fool all polynomial time algorithms with the same PRG. As we saw earlier this is not possible for all functions (not only computable by polynomial time algorithms), because the output of G is not really uniform.

Definition 1 *The PRG $G : \{0,1\}^s \rightarrow \{0,1\}^r$* fools all polynomial time algorithms, *if for some $\varepsilon > 0$, for every polynomial time algorithm $A(x, y)$, where $|x| = n$ and $|y| = r = n^{O(1)}$ and for every x*

$$|\Pr_{z \in_{\mathcal{U}} \{0,1\}^s}[A(x, G(z)) = 1] - \Pr_{y \in_{\mathcal{U}} \{0,1\}^r}[A(x, y) = 1]| < \varepsilon$$

We would like to have $\varepsilon = 1/n^{\Theta(1)}$, but for derandomization of **BPP** even $\varepsilon = 1/4$ is enough. Also if there is a polynomial-time computable PRG with $s = O(\log r) = O(\log n)$ then **BPP** = **P**. To get this run the randomized algorithm that has to be derandomized on the output of G for every seed and take majority vote of the results. In order for this derandomization to work G has to work in polynomial time. At the moment it is not known if such efficient PRG exists (and thus **BPP** = **P** is an open question), but we can prove that a PRG with the given above parameters and unknown complexity exists.

1.3 Circuits

Definition 2 A Circuit *is a directed acyclic graph, in which every node is marked as input, gate or output. The inputs have in-degree 0, the outputs have out-degree 0. The gates can be OR, AND or NOT and their in-degree is 2 for the OR and AND gates and 1 for the NOT gates. The out-degree of the inputs and the gates is unrestrained. The* size *of a circuit is the number of gates in it.*

If a circuit C has n input nodes and m output nodes than it naturally corresponds to a boolean function from $\{0,1\}^n$ to $\{0,1\}^m$. We will denote both the circuit and the function it computes with the same name. We will be concerned with circuits which have only one output.

Definition 3 A family of circuits *is a collection of circuits $\{C_n\}_{n \geq 0}$. A family of circuits has size $S(n)$ if the size of C_n is $S(n)$, for every n. The family is uniform if there is a Turing Machine which generates the circuits of the family in time polynomial in n and non-uniform otherwise.*

Families of circuits naturally correspond to functions from $\{0,1\}^*$ to $\{0,1\}$. The issue of uniformity is quite important because it is easy to see that there are non-uniform families of circuits of constant size which compute undecidable functions (for every input length n consider the circuit with constant output which is exactly the value of some undecidable function (say the characteristic function of the Halting Problem) on n). On the other hand it can be proved that the set of functions computable by a uniform family of polynomial size is exactly the set of functions for which there is a polynomial time Turing Machine.

We will strengthen the definition of a PRG and require it to fool non-uniform families of circuits.

Definition 4 *The PRG $G : \{0,1\}^s \rightarrow \{0,1\}^r$* fools all non-uniform families of polynomial size circuits, *if for some $\varepsilon > 0$, for every non-uniform family of polynomial size circuits $\{C_r\}_{r \geq 0}$*

$$|\Pr_{z \in_{\mathcal{U}} \{0,1\}^s}[C_r(G(z)) = 1] - \Pr_{y \in_{\mathcal{U}} \{0,1\}^r}[C_r(y) = 1]| < \varepsilon$$

This will allow us to fix the input x in the polynomial time randomized algorithm A that we want to fool and thus to obtain an algorithm A_x which depends only on the random bits. For every such A_x we can construct a polynomial size circuit. This way we obtain a family of polynomial size circuits $\mathcal{C}$. Now using a PRG that fools all non-uniform families of circuits we will be able to fool A as well (or at least $\mathcal{C}$, but this is the same).

1.4 Existence of PRG. Probabilistic Method

We will prove that a PRG with $s(r) = O(\log r)$ and $\varepsilon(r) = 1/r^{\Theta(1)}$ exists. The proof will involve a general technique known as probabilistic method. The probabilistic method is generally used to prove the existence of combinatorial objects. The essence of the method is constructing a suitable probabilistic space and then proving that the probability of sampling the combinatorial object of interest is non-zero, i.e. that such an object really exists. The probabilistic method is non-constructive; it only proves the existence of the object and does not give a procedure for obtaining it. Therefore, although we will prove the existence of a PRG with the above parameters, we will not be able to tell how to construct it and what is it complexity.

Theorem 1 *A PRG with $s(r) = O(\log r)$ and $\varepsilon(r) = 1/r^{\Theta(1)}$ exists.*

To prove the theorem we will use a technical result useful for bounding the probability of deviation of a sum of independent random variables from the expectation of the sum.

Lemma 1 (Chernoff Bound) *Let $X_1, \ldots, X_n$ be independent random variables in $[0,1]$, $X = \sum_{i=1}^n X_i$ and $\alpha > 0$. Then*

$$\Pr[X - E[X] \geq \alpha n] \leq e^{-\alpha^2 n/2}$$

By two application of the Lemma 1 we obtain.

Corollary 1 *Let $X_1, \ldots, X_n$ be independent random variables in $[0,1]$, $X = \sum_{i=1}^n X_i$ and $\alpha > 0$. Then*

$$\Pr[|X - E[X]| \geq \alpha n] \leq 2e^{-\alpha^2 n/2}$$

The above results show that the sum of independent random variables is not close to its expectation with exponentially small, in the number of random variables, probability.

Proof:[Theorem 1] Choose $T \subseteq \{0,1\}^r$ of size $t = poly(r) = 2^s$ randomly, i.e. pick independently, uniformly and with replacement t elements of $\{0,1\}^r$ and form a set with them. For every small circuit C we will prove that with non-zero probability over T the probability that C accepts a random string from T is close to the probability that C accepts a random string from $\{0,1\}^r$. Taking $range(G) = T$ will prove the theorem.

Let $C_1, \ldots, C_N$ be all circuits of size r^c, for some constant $c > 0$. Let $S_i = \{y \mid C_i(y) = 1\}$, for every $i \in [N]$. We have that

$$\begin{aligned}
\Pr_T[T \text{ does not yield a PRG}] &= \Pr_T\left[\exists i \in [N] \left| \frac{|T \cap S_i|}{t} - \frac{|S_i|}{2^r} \right| \geq \varepsilon \right] \\
&\leq \sum_{i=1}^{N} \Pr_T\left[\left| \frac{|T \cap S_i|}{t} - \frac{|S_i|}{2^r} \right| \geq \varepsilon \right] \\
&\leq N 2e^{-\varepsilon^2 t/2}
\end{aligned}$$

The first inequality comes from the union bound ($\Pr[A \cup B] \leq \Pr[A] + \Pr[B]$). Let us see that the second inequality holds.

Let $T = \{z_1, \ldots, z_t\}$. Consider S_i. Define the random variable Z_j to be 1, if $z_j \in S_i$, and 0 otherwise. Then $|T \cap S_i| = \sum_{j=1}^t Z_j =: Z$. On the other hand $E[Z_j] = |S_i|/2^r$, since T is a random set. From this follows that $E[Z] = t|S_i|/2^r$. Since $Z_1, \ldots, Z_t$ are independent by Corollary 1 we obtain that $\Pr_T[|Z - E[Z]| \geq \varepsilon t] \leq 2e^{-\varepsilon^2 t/2}$. Hence

$$\Pr_T\left[\left|\frac{|T \cap S_i|}{t} - \frac{|S_i|}{2^r}\right| \geq \varepsilon\right] \leq 2e^{-\varepsilon^2 t/2}$$

and the upper bound on the probability that T does not yield a PRG follows.

Consider now the expression $2Ne^{-\varepsilon^2 t/2} < e^{1+\ln N - \varepsilon^2 t/2}$. It is less than 1 for $1+\ln N - \varepsilon^2 t/2 < 0$, i.e. $t > 2(1+\ln N)/\varepsilon^2$. Let us estimate N from above.

Since for the type of every gate of a circuit of size l there are 3 possibilities and every gate takes input from at most 2 other gates there are at most $(3l^2)^l$ circuits with size l. Hence, since in our case $l = r^c$, for some constant c, $\ln N = O(r^c \log r)$.

Therefore for every constant $c' > 0$, such that $\varepsilon \leq 1/r^{c'}$, we can choose suitable constant $c'' > 0$, such that $t = r^{c''} > 2(1+\ln N)/\varepsilon^2$. Hence for $s = O(\log r)$ and $\varepsilon = 1/r^{\Theta(1)}$ the probability that T does not yield a PRG is less than 1, i.e. the probability that it does yield a PRG is greater than 0. From this it follows that there exists T yielding a PRG with the above parameters.

□

As we saw earlier if a PRG with $s = O(\log n)$ exists, we can derandomize completely all polynomial time randomized algorithms. Since the existence of a PRG given by Theorem 1 is non-constructive, we can derandomize all polynomial time randomized algorithms by non-uniform families of polynomial size circuits. The construction of the circuit family stays the same (we try the algorithm on all outputs of the PRG and then take majority); the non-uniformity appears because we know only about the existence of small sets of pseudorandom strings on which to run the algorithm. Formally we have

Corollary 2 *Every language in* **BPP** *has a non-uniform family of polynomial size circuits computing it.*