A pseudorandom generator is an efficient deterministic algorithm or circuit that takes a short uniformly random seed as its input and produces a longer output that looks indistinguishable from a uniformly random string of the same length to all efficient "adversaries" that do not know the seed.

Definition 1. Two distributions X and Y are *ε-indistinguishable* by size-S circuits if for all circuits of size at most S, $|\Pr[D(X) = 1] - \Pr[D(Y) = 1]| < \varepsilon$.

Definition 2. A function $G : \{0,1\}^k \to \{0,1\}^m$, where $k < m$ is an *ε-pseudorandom generator* against size-S circuits if the distribution $G(s)$ where $s \sim B^k$ is uniform is ε-indistinguishable from the uniform distribution on $\{0,1\}^m$ by size S circuits.

The circuit D is called a *distinguisher*, and the difference between the two probabilities is called the *advantage* of D. Since the output of a pseudorandom generator is longer than its input, its output is *statistically* distinguishable from a uniformly random string: The distinguisher that outputs 1 on input y if $y = G(s)$ for some s and 0 otherwise has advantage at least $1 - 2^{k-m}$. A brute-force implementation of this distinguisher would need to check whether y equals $g(s)$ for any one of the 2^k strings y, thereby taking up exponential (in k) size. The gold standard in the design of pseudorandom generators is to rule out distinguishers that are substantially better than brute force.

The notion of efficient indistinguishability is a very strong one: It implies that no single bit is substantially biased, no pair of bits are substantially correlated, the majority of any odd number of bits is close to unbiased, if the string is interpreted as the adjacency matrix of the graph then the graph has no sparse cut, and so on, as all these conditions can be verified by efficient distinguishers.

There are two types of pseudorandom generators depending of the relative computational power of the generator and the distinguisher. Pseudorandom generators that are more complex than their distinguishers have applications to the deterministic simulation of randomized algorithms and proofs. Generators that are pseudorandom even against distinguishers of higher relative complexity are a central object in cryptography.

In the last lecture we saw an effective tool for arguing that some NP-decision and search problems are unlikely to admit much faster algorithms than brute-force search: NP-completeness. We might therefore expect that NP-completeness should play a role in the design of pseudorandom generators. One important difference is that Definition 2 is required to hold for a *random* choice of input s, while NP-hardness only guarantees hardness against worst-case inputs. In contrast, pseudorandom generators are related to questions about *average-case* hardness.

1 Simulating randomness

Here is a strategy for deterministically simulating efficient randomized algorithms for decision problems, such as the algorithm for polynomial identity testing. Suppose that we have a decision problem f, a randomized algorithm A, and an input $x \in \{0,1\}^n$ such that A solves f on x with a clear majority, namely

$$\Pr_{r\sim\{0,1\}^m}[A(x,r) = f(x)] \geq 2/3$$

where m is the amount of randomness used by the algorithm on inputs of length n. If A runs in time $t(n)$ on inputs of length n then there is a circuit C_x of size $O(t(n)^2)$ such that

$$\Pr_{r\sim\{0,1\}^m}[C_x(r) = f(x)] \geq 2/3.$$

Now if $G\colon \{0,1\}^k \to \{0,1\}^m$ was a, say, 1/6-pseudorandom generator against size $O(t(n)^2)$ circuits then

$$\begin{aligned}&\left|\Pr_{s\sim\{0,1\}^k}[C_x(G(s)) = f(x)] - \Pr_{r\sim\{0,1\}^m}[C_x(r) = f(x)]\right| \\ &\qquad = \left|\Pr_{s\sim\{0,1\}^k}[C_x(G(s)) = 1] - \Pr_{r\sim\{0,1\}^m}[C_x(r) = 1]\right| < 1/6\end{aligned}$$

so in particular

$$\Pr_{s\sim\{0,1\}^k}[A(x,G(s))=f(x)]=\Pr_{s\sim\{0,1\}^k}[C_x(G(s))=f(x)]>2/3-1/6=1/2.$$

We can now simulate A on input x deterministically by enumerating all possible outputs of $G(s)$ and observing what fraction of the time $A(x,G(s))$ accepts. If A were to accept, a majority of the outputs $G(s)$ should yield accepting computations; if A were to reject, the majority of them should yield rejecting computations. If the output of G can be computed in time t' then the simulation takes time $O(2^k(t(n)+t'(k)))$. For a polynomial-time algorithm A both the size $O(t(n)^2)$ and the number of random bits m (which is upper bounded by the size) is polynomial in n. If k grows at most logarithmically in m and $t'(k)$ is at most exponential, the whole simulation can be carried out in polynomial time implying that P = BPP.

Proposition 3. *If there exists a constant C such that for all m there is a $1/6$-pseudorandom generator against size m with seed length $k=C\log m$ and output length m that is computable in time $2^{Ck}=m^C$ (uniformly for all m) then* P = BPP.

2 Distinguishing versus predicting

One criticism of the theory of NP-completeness that we saw last time is that it measures computational hardness in terms of the *worst-case* running time of algorithms over all inputs of a given size. Sometimes it is more relevant how algorithms behave on "typical" inputs. Average-case complexity assumes that inputs come from some probability distribution and algorithms are allowed to err with some small probability over the choice of input. Two types of algorithms that fit nicely into this setting are learning algorithms (whose inputs consist of training examples, which are sometimes modeled as independent samples from some distribution) and cryptographic adversaries (whose success is determined by how likely they are to break a random execution of the scheme).

In average-case complexity the inputs come from some distribution $\mathcal{D}$ and we are interested in computing the result correctly for typical inputs sampled from $\mathcal{D}$. Let f be a function from $\{0,1\}^n\to\{0,1\}$.

Definition 4. An *predictor* for f with advantage ϵ under distribution $\mathcal{D}$ is a circuit P for which $\Pr[P(x)=f(x)]\geq(1+\varepsilon)/2$ when x is sampled from $\mathcal{D}$.

To get a sense of the definition imagine $\mathcal{D}$ is a distribution on images x, half of which are cats and half of which are dogs. The goal of the predictor P is to guess the true label of the image (0 for cat, 1 for dog). Advantage 1 means perfectly accurate prediction, which is the same as deciding f. Advantage zero can be achieved without training by always guessing "cat"; in general, one of the constants $0,1$ always has nonnegative advantage.

A function is ε-unpredictable against size S if there does not exists a size S-predictor for f with advantage at ε. The examples that we say in lectures 1-4 ($PARITY$ for constant-depth AND/OR circuits, $MAJORITY$ for constant depth AND/OR/PARITY circuits, IP for read-once branching programs) are not only hard to compute but also ε-unpredictable for any constant $\varepsilon>0$ as the input length tends to infinity.

As for general circuits, it can be shown by a counting argument that at most a $2^{2^{\delta n}}$ fraction of all functions are predicatable with advantage $2^{-\delta n}$ by circuits of size $2^{\delta n}$ for some constant $\delta>0$ (I think $\delta=1/8$ is good enough). So functions that are very hard to predict certainly exist. We would be hard pressed to find any "explicit" functions, say within NP, that are hard to predict, because proving a function is hard to predict is only harder than proving it is hard to compute, which would resolve the P $\neq$ NP question. Despite the lack of provably hard candidates there are very concrete examples of functions that are believed to be hard to predict. We'll say more about this shortly.

One useful insight of average-case complexity is that distinguishing and prediction are equivalent in the following sense. A predictor P for f has the ability to distinguish the distribution $(x,f(x))$ from (x,b) where b is a uniformly random bit that is independent of x. To distinguish, simply apply P to the x-part of the input and accept if the second part (call it y) equals $P(x)$. If the input is distributed like $(x, f(x))$ then the distinguisher accepts precisely with probability $\Pr[P(x) = f(x)] = (1+\varepsilon)/2$. If the input is distributed like (x, b) then the distinguisher accepts with probability half because b is independent of the prediction. Therefore ε-predictors of size S can be used to construct $\varepsilon/2$-distinguishers of size $S + O(1)$. The converse is also true:

Lemma 5 (Yao's lemma)**.** *If $(x, f(x))$ is distinguishable from (x, b) (where b is a random bit independent of x) with advantage ε by a size-S circuit then f is predictable with advantage 2ε by a circuit of size $2S + O(1)$.*[1]

Stated in contrapositive form we obtain the following important corollary:

Corollary 6. *If $f\colon \{0,1\}^k \to \{0,1\}$ is ε-unpredictable against size S with respect to the uniform distribution then the function $G : \{0,1\}^k \to \{0,1\}^{k+1}$ given by $G(s) = (s, f(s))$ is 2ε-pseudorandom against size $S/2 - O(1)$.*

Thus unpredictability, i.e., average-case hardness for decision problems, can be converted into pseudorandomness.

Proof of Yao's lemma. We prove the contrapositive. Assume D is a circuit such that

$$\big|\Pr[D(x, f(x)) = 1] - \Pr[D(x, b) = 1]\big| \geq \varepsilon,$$

Using D we want to construct another circuit P that predicts f with advantage ε. For this it will be first convenient to get rid of the absolute value. By possibly replacing the circuit D by NOT D, we may assume without loss of generality that

$$\Pr[D(x, f(x)) = 1] - \Pr[D(x, b) = 1] \geq \varepsilon. \tag{1}$$

Therefore, on average over the choice of x, the advantage $\varepsilon(x)$ of the distinguisher is at least ε:

$$\mathrm{E}[\varepsilon(x)] \geq \varepsilon, \qquad \text{where } \varepsilon(x) = \Pr[D(x, f(x)) = 1 \mid x] - \Pr[D(x, b) = 1 \mid x].$$

For a fixed choice of x, the distinguisher $D(x, y)$ computes one of four possible functions of y: the constant 0 or 1, the identity y, or negation NOT y. In the first two cases $\varepsilon(x)$ must be zero, so a predictor that outputs a random bit has conditional advantage $\Pr[P(x) = f(x) \mid x] = 1/2 = 1/2 + \varepsilon(x)$. If $D(x, \cdot)$ is the identity then $f(x)$ equals one with probability $1/2 + \varepsilon(x)$, so a predictor that outputs 1 has this advantage given x. If $D(x, \cdot)$ is negation then $f(x)$ equals zero with probability $1/2 + \varepsilon(x)$, so a predictor that outputs 0 has this advantage given x. In summary, the advantage of the predictor

```
$P$ :  On input $x$,
       Learn the function $D(x, \cdot)$ by querying $D(x, 0)$ and $D(x, 1)$.
       If $D(x, \cdot)$ is a constant, output a random bit.
       If $D(x, \cdot)$ is the identity, output 1.
       If $D(x, \cdot)$ is negation, output 0.
```

is at least $1/2 + \varepsilon(x)$ given x, so at least $\mathrm{E}[1/2 + \varepsilon(x)] = 1/2 + \varepsilon$ on average. □

[1] There is a different proof that gives advantage ε for size $S + O(1)$.

3 The Nisan-Wigderson generator

One weakness of Corollary 6 is that it yields only one bit of pseurodrandomness beyond the "entropy" of the seed s. To make use of Proposition 3 we need exponentially many pseudorandom bits. Those are provided by the Nisan-Wigderson generator. We will say that a family $G_m\colon \{0,1\}^{k(m)} \to \{0,1\}^m$ of pseudorandom generators is polynomial-time computable if there is an algorithm that on input $s \in \{0,1\}^{k(m)}$ runs in time polynomial in m (the *output length* of G_m) and outputs $G_m(s)$.

Theorem 7 (The Nisan-Widgerson generator)**.** *For every polynomial S and every constant $\delta > 0$ the following holds. Suppose there is a decision problem f that can be decided in time $2^{O(t)}$ on all inputs of length t, but is unpredictable with advantage $2^{-\delta t/2}$ by circuits of size $O(2^{\delta t})$ with respect to the uniform distribution over $\{0,1\}^t$. Then there exists a polynomial-time computable family $G_m\colon \{0,1\}^{k(m)} \to \{0,1\}^m$ of $1/6$-pseudorandom generators against circuits of size at most $m = S(n)$.*

Such hard problems f are believed to exist, but I don't know any natural candidate examples. The usual interpretation of Theorem 7 is that it provides evidence for P = BPP. In particular, it supports the belief that specific problems in BPP like polynomial identity testing should have efficient deterministic algorithms, though the search for a concrete algorithm that is provably correct on all inputs without additional assumptions is still ongoing.

Towards proving Theorem 7, let us first see how to get *two* additional bits of pseudorandomness from Lemma 5: We run the generator on two independent seeds s_1 and s_2. Namely, we let $G'(s_1, s_2) = (G(s_1), G(s_2))$. If D is a distinguisher such that

$$\left|\Pr[D(G'(s_1, s_2)) = 1] - \Pr[D(y_1, y_2) = 1]\right| \geq \varepsilon$$

then it must be the case that

$$\begin{aligned} &\left|\Pr[D(G(s_1), G(s_2)) = 1] - \Pr[D(G(s_1), y_2) = 1]\right| \geq \varepsilon/2 \\ \text{or} \quad &\left|\Pr[D(G(s_1), y_2) = 1] - \Pr[D(y_1, y_2) = 1]\right| \geq \varepsilon/2 \end{aligned}$$

and in either case we can obtain a distinguisher D' such that

$$\left|\Pr[D'(G(s)) = 1] - \Pr[D'(y) = 1]\right| \geq 1/2$$

by hardwiring the suitable input that maximizes the advantage into D. By repeating this construction using seeds $s_1, \ldots, s_t$ we can obtain $t(k+1)$ pseudorandom bits out of a seed of length tk as long as t is not too large (as the distinguishing advantage deteriorates by a factor of t).

To further shrink the seed length, we will allow parts of the strings $s_1, \ldots, s_t$ to overlap. This motivates the following definition.

Definition 8. A collection of sets $T_1, \ldots, T_m \subseteq \{1, \ldots, k\}$ is a *combinatorial design* with set size t and intersection size $t_\cap$ if $|T_i| = t$ for every i and $|T_i \cap T_j| \leq t_\cap$ for every $i \neq j$.

Given a combinatorial design, we define a pseudorandom generator $G : \{0,1\}^k \to \{0,1\}^m$ by

$$G(s) = (f(s|_{T_1}), \ldots, f(s|_{T_m}))$$

where $f : \{0,1\}^t \to \{0,1\}$ is the "hard" function and $s|_T$ is the substring of s indexed by the elements of the set T. For example, if $s = s_1s_2s_3s_4$, then $s|_{\{2,4\}} = s_2s_4$. Combinatorial designs with good parameters can be computed efficiently.

Claim 9. *For every $c > 0$ there is a family of combinatorial designs with*

$$k = 6c^2 \log m \qquad t = c \log m \qquad t_\cap = \log m.$$

Moreover, there is a deterministic algorithm that on input m runs in time $m^{O(c^2)}$ and outputs the sets $T_1, \ldots, T_m$.

In particular, for every fixed c we have $k = O(\log m)$, so the seed size is exactly what we were aiming for. Let's now check that G can be computed efficiently (in time $\mathrm{poly}(m)$). To compute G_m, we first construct the design in time $m^{O(c^2)}$. We then need to evaluate m copies of f, each on an input of size $t = c \log m$. Since we assumed that f is computable in time $2^{O(t)} = m^{O(c)}$, the whole computation can be done in time polynomial in m.

It remains to show that G_m is 1/6-pseudorandom against circuits of every polynomial size $S(n)$ for a suitable choice of constants c and δ. Towards a contradiction, suppose that for some distinguisher circuit D of size $S(n)$ we have

$$\Pr_{s\sim\{0,1\}^k}[D(G(s)) = 1] - \Pr_{r\sim\{0,1\}^m}[D(r) = 1] \geq 1/6.$$

(As in the last proof, we can remove the absolute value without loss of generality.) Let's expand this definition:

$$\Pr_{s\sim\{0,1\}^k}[D(f(s|_{T_1}), \dots, f(s|_{T_m})) = 1] - \Pr_{r_1,\dots,r_m\sim\{0,1\}}[D(r_1, \dots, r_m) = 1] \geq 1/6. \tag{2}$$

This formula does not appear all that useful. To see what is happening, we introduce of the following way of "slowly" going from the pseudorandom distribution $G(s)$ to the random distribution r: At each step, we change one input of D from pseudorandom to random. If D can distinguish $G(s)$ from r, then at some step there must be a noticeable change in the behavior of C.

More formally, we consider the following sequence of *hybrid distributions* on inputs of D:

$$\begin{array}{rcccc}
H_m: & f(s|_{T_1}), & \dots, & f(s|_{T_{m-1}}), & f(s|_{T_m}) \\
H_{m-1}: & f(s|_{T_1}), & \dots, & f(s|_{T_{m-1}}), & r_m \\
 & \vdots & & \vdots & \vdots \\
H_0: & r_1, & \dots, & r_{m-1}, & r_m.
\end{array}$$

Condition (2) tells us that $\Pr_{r\sim H_m}[D(r)] - \Pr_{r\sim H_0}[D(r)] \geq 1/6$. Then there must be some j between 1 and m for which $\Pr_{r\sim H_j}[D(r)] - \Pr_{r\sim H_{j-1}}[D(r)] \geq 1/6m$, that is

$$\begin{aligned}\Pr_{s\sim\{0,1\}^k, r_{j+1},\dots,r_m\sim\{0,1\}}&[D(f(s|_{T_1}), \dots, f(s|_{T_j}), \dots, r_m) = 1] \\ &- \Pr_{s\sim\{0,1\}^k, r_j,\dots,r_m\sim\{0,1\}}[D(f(s|_{T_1}), \dots, r_j, \dots, r_m) = 1] \geq 1/6m.\end{aligned}$$

There must then exist a fixing of the values $r_{j+1}, \dots, r_m$ that maximizes the above difference in probabilities. As those are independent on s conditioning on this choice does not change the distribution on s. If we hardwire this fixing into the circuit D, we obtain a circuit D_1 of the same size such that

$$\Pr_s[D_1(f(s|_{T_1}), \dots, f(s|_{T_{j-1}}), f(s|_{T_j})) = 1] - \Pr_{s,r_j}[D_1(f(s|_{T_1}), \dots, f(s|_{T_{j-1}}), r_j) = 1] \geq 1/6m.$$

Let $s' = s|_{T_j}$ (this is a string of length t). There is now a fixing of all the bits of s outside s' that maximizes the above difference in probabilities. Let's hardwire these bits into D_1 and call the resulting circuit D_2. With respect to this fixing, for every $i < j$, $f(s|_{T_i})$ becomes a function of at most $\log m$ bits in s' (because s' intersects $s|_{T_i}$ in at most $t_\cap = \log m$ positions). Let's call this function $g_i(s')$. We then have

$$\Pr_{s'}[D_2(g_1(s'), \dots, g_{j-1}(s'), f(s')) = 1] - \Pr_{s',r_j}[D_2(g_1(s'), \dots, g_{j-1}(s'), r_j) = 1] \geq 1/6m.$$

Since each g_i is a function of at most $\log m$ bits, it can be computed by a circuit of size $O(2^{\log m}) = O(m)$. If we compose the circuit D_2 with the circuits for $g_1, \dots, g_{j-1}$, we obtain a single circuit D_3 of size $S(n) + O(jm) = S(n) + O(m^2)$ such that

$$\Pr_{s'}[D_3(s', f(s')) = 1] - \Pr_{s',r_j}[D_3(s', r_j) = 1] \geq 1/6m.$$

In words, $(s', f(s'))$ is not $1/6m$-pseudorandom for size $S(n)+O(m^2)$. By Yao's Lemma there is a predictor circuit P of size $2S(n)+O(m^2)$ such that

$$\mathrm{Pr}_{s'}[P(s') = f(s')] \geq 1/2 + 1/3m.$$

As f is a function on $t = c\log m$ bits, the advantage of P in terms of t is $\Omega(2^{-t/c})$ and its size is $2S(n)+O(m^2) = O(2^{2t/c})$. Choosing $\delta = 1/2c$ matches the claimed parameters and proves the theorem.

Proof of Claim 9. The sets $T_1, \dots, T_m$ are chosen greedily: T_i is the first set of size t that has intersection size at most $t_\cap$ with all sets $T_j, j < i$. The running time is dominated by the number of possible choices for each set which is at most $2^k = m^{O(c^2)}$.

We show that a choice of T_i with the desired properties is always possible by the probabilistic method. The probability that the intersection between the random set T_i and any fixed set T_j, $j < i$, both of size t, exceeds size $t_\cap$ is at most the probability that T_i contains some S subset S of T_j of size $t_\cap$ that is already contained in T_j. The probability that T_i contains a fixed such S equals $t/k \cdot (t-1)/(k-1) \cdots (t-t_\cap+1)/(k-t_\cap+1) \leq (t/k)^{t_\cap}$. As there are $\binom{t}{t_\cap}$ choices for S inside T_j, by a union bound

$$\Pr\bigl[|T_i \cap T_j| \geq t_\cap\bigr] \leq \binom{t}{t_\cap} \cdot \Bigl(\frac{t}{k}\Bigr)^{t_\cap} \leq \Bigl(\frac{et}{t_\cap}\Bigr)^{t_\cap} \cdot \Bigl(\frac{t}{k}\Bigr)^{t_\cap} = \Bigl(\frac{et^2}{t_\cap k}\Bigr)^{t_\cap} \leq 2^{-\log m} = \frac{1}{m}.$$

By another union bound over j, there must always exists a choice of T_i that has intersection less than $t_\cap$ with T_1 up to T_{i-1}. $\square$

4 Cryptographic pseudorandom generators

In the constructions from the previous two sections the time that it takes to compute the pseudorandom generator G is inherently larger than the time it takes to evaluate the hard function f as G must evaluate one or several copies of f. Therefore the complexity of computing G will be larger than the complexity of distinguishing the output of G from a uniformly random string.

In cryptographic applications the pseudorandom generator is usually part of the system and the distinguisher is the adversary that wants to break the system. It is usually assumed that the adversary is willing to invest more resources into breaking the system than the honest parties use to run it. In this setting generating even one additional bit of pseudorandomness beyond the seed length is challenging.

Cryptographic pseudorandom generators can be obtained from a simpler object called a one-way function. A one-way function is a function with multi-bit output that is easy to compute in the worst-case, but hard to invert even on average.

Definition 10. A function $f\colon \{0,1\}^n \to \{0,1\}^m$ is (S,ε)-one way if for every circuit I of size at most S, $\mathrm{Pr}_{x\sim\{0,1\}^n}[f(I(f(x)))) = f(x)] \leq \varepsilon$.

If P equals NP then the NP-search problem "given y find x such that $y = f(x)$" can be efficiently solved, so proving the existence of one-way functions requires proving that P does not equal NP. There are many conjectured examples of one-way functions, some with very simple structure. For example, if $m = n$ and every output bit of f is the majority of five randomly chosen input bits, f might be one-way with high probability. Cryptographic pseudorandom generators can be obtained from one-way functions:

Theorem 11. *For every function $f\colon \{0,1\}^n \to \{0,1\}^m$ computable by a circuit of size s there exists a function $G\colon \{0,1\}^k \to \{0,1\}^{k+1}$ computable by a circuit of size polynomial in s and n so that the following holds: If f is (S,ε)-one-way then G is ε'-pseudorandom against size S' circuits, where $S' = S - \mathrm{poly}(s,n)$ and $\varepsilon' = \varepsilon \cdot \mathrm{poly}(s,n)$.*

Once one additional bit of pseudorandomness is obtained, it is possible to increase the length of the output by applying the pseudorandom generator iteratively. Specifically, if $G\colon \{0,1\}^k \to \{0,1\}^{k+1}$ is a pseudorandom generator, then we iteratively define $G_0(s) = s$ and

$$G_{d+1}(s) = (G(\text{first } k \text{ bits of } G_d(s)), \text{last } d \text{ bits of } G_d(s)).$$

Then we can prove the following by induction on d.

Lemma 12. *If G can be computed by a circuit of size s and G is ε-pseudorandom against size S circuits then G_d is $d\varepsilon$-pseudorandom against size $S-(d-1)s$ circuits.*

Proof. For contradiction, let us suppose that G_{d+1} is not pseudorandom. So there exists a circuit D of size $S - ds$ such that

$$|\Pr[D(G_{d+1}(s)) = 1] - \Pr[D(y) = 1]| \geq (d+1)\varepsilon.$$

Here, Y is a truly random string of length $k+d+1$. Now recall that G_{d+1} was obtained by running G on the first k bits of y_d (the output of G_d) and copying the last d bits. Let $x_d = x_L x_R$, where X_L are the first k bits and y_R are the last d. Also let $y = y_L y_R$ where y_L are the first $k+1$ bits and Y_R are the last d bits. Then

$$|\Pr[D(G(x_L), x_R) = 1] - \Pr[D(y_L, y_R) = 1]| \geq (d+1)\varepsilon.$$

Now let z be a uniform string of length k independent of y. At least one of these two inequalities must hold:

$$|\Pr[D(G(x_L), x_R) = 1] - \Pr[D(G(z), y_R) = 1]| \geq d\varepsilon \qquad \text{or}$$
$$|\Pr[D(G(z), y_R) = 1] - \Pr[D(y_L, y_R) = 1]| \geq \varepsilon.$$

Suppose the first inequality holds. Then we can distinguish x from a truly random string as follows:

Circuit D': On input u, write $u = u_L u_R$ (the first k and last d bits) and output $D(G(u_L), u_R)$.

This circuit D' has size $S-(d-1)s$ and by the first inequality

$$|\Pr[D'(x_L, x_R) = 1] - \Pr[D'(z, y_R) = 1]| = |\Pr[D(G(x_L), x_R) = 1] - \Pr[D(G(z), y_R) = 1]| \geq d\varepsilon$$

so D' distinguishes the output of $G_d(s)$ from a random string with advantage $d\varepsilon$, contradicting our inductive assumption. So the second inequality must hold. But then we can distinguish the output of G from random by the following circuit: On input u, choose a random string y_R of length d and output $D(u, y)$. By the second inequality, this is a circuit of size $S - ds + d \leq S$ that distinguishes the output of G from random with advantage ε, contradicting the pseudorandomness of G. □