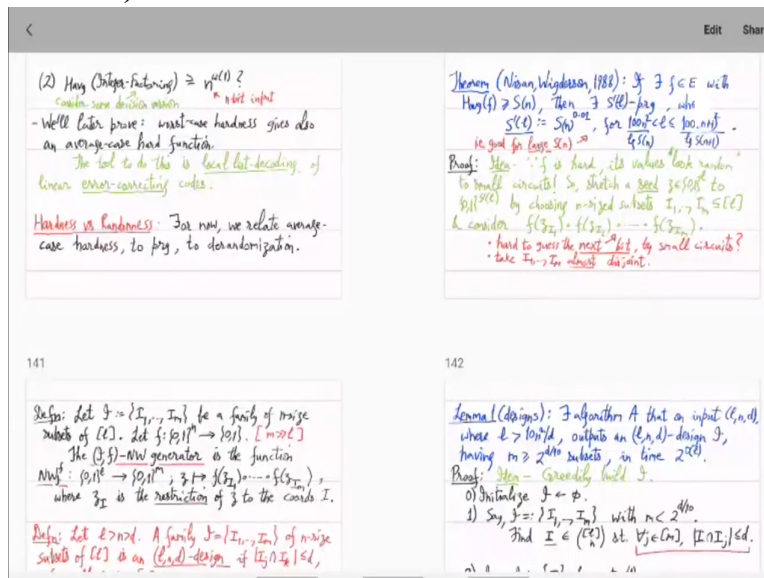


Randomized Methods in Complexity
Prof. Nitin Saxena
Department of Computer Science and Engineering
Indian Institute of Technology - Kanpur

Lecture - 19
Hardness to NW- Generator to PRG

(Refer Slide Time: 00:15)



So, in the previous class we started this theorem. It is a very famous theorem in computer science due to Nisan and Wigderson. It shows that if there is an explicit function f which is average-case hard then you can use that to stretch a seed and get a pseudorandom string. So, basically, you can use average-case hardness to get a prg. So, $S(n)$ hardness will give you $S'(n)$ prg. So, this will prove in many steps it will be a long proof first.

So, the idea of this is that use the average-case hard function f on our design. Design means that you take a seed, seed string is z and on substrings of z you evaluate f and these bits become your output this is the stretched seed. So, first we have to show that this design exists that was lemma 1 which we have shown by a greedy analysis. Second thing we have to show is that if this string based on design \mathcal{I} and f if this string is not pseudorandom then f will be average-case easy.

Basically the idea there is that we will construct a bit predictor so that is lemma 2. Here we want to show that using the design \mathcal{I} and f this Nisan Wigderson (\mathcal{I}, f) image on U_1 . This is pseudorandom and the idea here is that if it is not then there will be a distinguisher circuit C which we can use to predict bits.

(Refer Slide Time: 02:20)

- We use the design in (I, f) -NW generator now.

Lemma 2 (NW-generator): If I is an (L, n, d) -design with $|I| = 2^{d/10} =: m$; $f: \{0,1\}^n \rightarrow \{0,1\}$ with $H_{\text{avg}}(f) > 2^d$, then $NW_I^f(U_e)$ is $(H_{\text{avg}}(f)/10, 0.1)$ -pseudorandom.

Proof: Idea - Suppose circuit C "distinguishes" $NW(U_e)$ from U_m . Then, we'll design a bit-predictor circuit C' for $f(z_{I_i})$, for some $i \in [m]$. C' will contradict the avg-case hardness of f !

And hence f will become easy which is a contradiction. So that is the idea we want to construct a bit predictor circuit C' and that will contradict the average case hardness of f .

(Refer Slide Time: 02:32)

• Let $S := H_{\text{avg}}(f)$.

• Suppose \exists circuit C of size $\leq S/10$ st.
 $|\Pr[C(NW_I^f(U_e))=1] - \Pr[C(U_m)=1]| \geq 0.1$.
 (ie $NW(U_e)$ is not pseudorandom) \rightarrow

• Wlog, assume $\Pr[C(NW(U_e))=1] - \Pr[C(U_m)=1] \geq 0.1$.

• We'll now devise a bit-predictor for NW_I^f .

• Identify the bit which can be predicted:

• For that, define distributions D_0, \dots, D_m over $\{0,1\}^n$ s.t.

$\forall i, D_i$: • choose $x \in U_e$; $z_{i+1}, \dots, z_m \in \{0,1\}^n$

• Compute $y := NW_I^f(x)$.

• Output $\langle y_1, \dots, y_i; z_{i+1}, \dots, z_m \rangle$.

Hybrid distribution \rightarrow

So, how do you construct C' from C is the next question. So, let S be the average-case hardness of f . And suppose this circuit see is the distinguisher it is small size $S/10$. And it is able to distinguish between this Nisan Wigderson (I, f) on U_1 distribution from the C of U_m distribution from NW distribution on U_1 to U_m that is what; remember stretch is from l to m that is the stretch following the design.

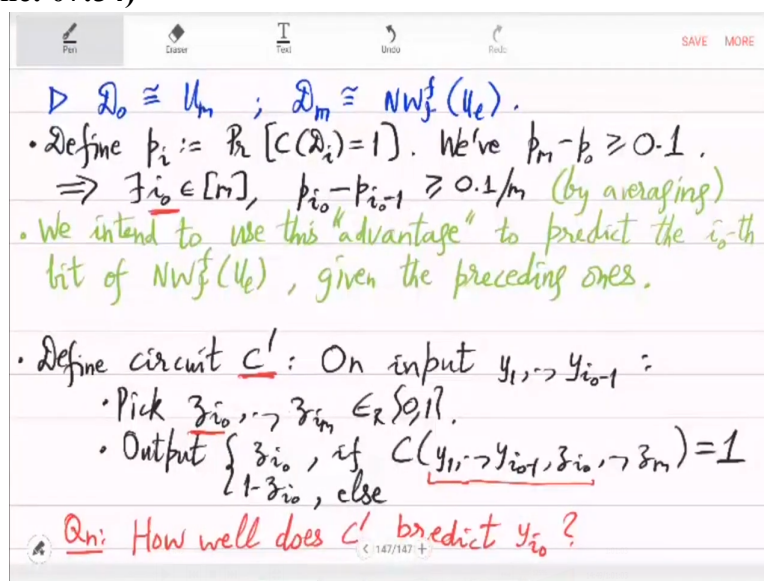
So, this difference in the probability is more than 0.1 let us assume that without loss of generality, we can assume that it is the difference is positive. So, the probability difference is at least 0.1 and let us now devise a bit predictor. So, the way we will create the bit predictor is

we will try to first locate the place where the bit prediction can happen. So, NW U_1 has m bits it is an m bit string we first have to identify the location where the prediction can happen.

Identify the bit which can be predicted so, let us first do that so define distributions D_0 to D_m such that for all i this distribution D_i will basically use only the prefix of NW distribution. Instead of using the whole output of NW you just use the first i bits and the remaining part you choose randomly that is the idea. So, D_i is first choose x from the distribution U_1 so $\{0,1\}^i$ and z_{i+1} to z_m R bits compute y that is the NW (I,f) string.

So, this we use the design I and the average case have function f it will give you m bits only look at the first i bits that is the new stretched string instead of y_1 to y_n you just output y_1 to y_i and the suffixes randomly chosen $m - i$ bits z_{i+1} to z_m . So, this distribution will call D_i and now you can look at the whole spread. So, D_0 to D_m these distributions D_0 is all the bits will be random D_1 is first is y_1 the remaining bits are purely random and D_m will be all the bits are y_1 to y_m . That is the whole spread of these distributions D_i that we have defined this is called hybrid distribution.

(Refer Slide Time: 07:34)



So, you can observe that D_0 is the uniform distribution. And as I said D_m will be your Nisan Wigderson NW. So, let us look at the way C behaves on these distributions so the probability of C on D_i being 1 remember C is this distinguisher circuit that we have assumed. So, probability of it on D_i is p_i and what do you know is that $p_m - p_0$ is greater than equal to 0.1 Because p_m is a Nisan Wigderson corresponds to NW distribution p_0 corresponds to the U_m distribution.

So, the difference of C on these 2 probability differences is large; it is 0.1. So, from that you can deduce that there is an i_0 such that $p_{i_0} - p_{i_0-1}$ is 0.1 by m this you get by averaging because for all i_0 these differences are small then the difference between p_m and p_0 will also be smaller than 0.1 that cannot happen. So, somewhere it has to be larger than this average 0.1 by m . So that now kind of gives me the location i_0 .

This is the bit where there seems to be some probability difference which is large. So, I think that this is the location this is the bit that is predictable. So, how do you predict it? So, you predicted by basically evaluating C if you get 1 then maybe you put some value for the bit otherwise you put the opposite. So, we will now work this out so we intend to use this advantage to predict the i_0 -th bit of $NW(I, f)$ and U_1 given the preceding ones.

So, if you are given everything till $i_0 - 1$ then the i_0 th we will try to predict using this probability difference and obviously using the circuit C that is given to us. So, let us define that predictor so on input y_1 to y_{i_0-1} pick z_{i_0} to z_{i_m} random bits. Now you have m bits prefixes y suffixes z and just see whether what C evaluates to on this. If it evaluates to 1 then you will output z_{i_0} . If it does not output 1 then you will negate z_{i_0} that is the idea of prediction.

Essentially C' is just guessing the i_0 th bit by randomly picking it and evaluating C . So, z_{i_0} if C on these y is and z suffix evaluates to 1. Why are we outputting z_{i_0} when C evaluates to 1? Well because you know that C evaluating to 1 is higher than there is this difference in the probability p_{i_0} is sufficiently bigger than p_{i_0-1} . And p_{i_0} was the probability of distribution D_{i_0} D_{i_0} .

Remember in D_{i_0} you are using the first i_0 bits from the NW generator. If you use NW generator based prefix then there is a higher chance of C output 1. So, hence I mean intuitively it makes sense to whenever C evaluates to 1 in this situation whenever you see with this prefix C is evaluating to 1 then there is a better chance of z_{i_0} being the correct answer.

So, we have to actually compute this probability but this is the insight behind the definition of C' otherwise you just negate z_{i_0} . So, obviously the question is how well does C' predict? y_{i_0}

what is the success probability? So, this will be kind of Biezen probability analysis so, let us do that.

(Refer Slide Time: 14:53)

The slide contains the following handwritten derivation:

$$\Rightarrow \Pr_{\substack{y \in \text{NW}(U_n) \\ z \in U_m}} [C'(y_1, \dots, y_{i_0-1}) = y_{i_0}] =$$

$$+ \Pr[z_{i_0} = y_{i_0}] \cdot \Pr[C(y_1, \dots, y_{i_0-1}, z_{i_0+1}, \dots, z_m) = 1 \mid z_{i_0} = y_{i_0}]$$

$$+ \Pr[z_{i_0} \neq y_{i_0}] \cdot \Pr[C(y_1, \dots, y_{i_0-1}, z_{i_0+1}, \dots, z_m) = 0 \mid z_{i_0} \neq y_{i_0}]$$

$$= \frac{1}{2} \cdot \Pr[C(D_{i_0}) = 1] + \frac{1}{2} \cdot (1 - \Pr[C(y_1, \dots, y_{i_0-1}, \bar{y}_{i_0}, z_{i_0+1}, \dots, z_m) = 1])$$

$$= p_{i_0} + \frac{1}{2} - \frac{1}{2} (p_{i_0} + \Pr[C(y_1, \dots, y_{i_0-1}, \bar{y}_{i_0}, z_{i_0+1}, \dots, z_m) = 1])$$

$$= p_{i_0} + \frac{1}{2} - \frac{1}{2} (2 \times p_{i_0-1}) = \frac{1}{2} + (p_{i_0} - p_{i_0-1}) \geq \frac{1}{2} + \frac{0.1}{m}$$

$\Rightarrow C'$ is a decent bit-predictor!

To make C' deterministic, fix z_{i_0+1}, \dots, z_m suitably; get circuit

The probability of C' given y_1 to y_{i_0-1} predicting y_{i_0} where y refers to the NW output and z refers to U_m . This probability y 's are picked from the prefix of NW and z 's are picked from the suffix of U_m . That is what we mean by this notation so this is equal to let us write down 2 cases. First is z_{i_0} being equal to y_{i_0} which means that z_{i_0} was the correct value. So, what is the probability of C' being y_{i_0} in that case.

In that case basically C has to evaluate to 1 plus probability that z_{i_0} is not y_{i_0} . In this case C' will output y_{i_0} only when C evaluates to 0 so, let us write that. This is equal to so z_{i_0} is randomly chosen so this probability is half times the second probability is that is just the probability of D_{i_0-1} . Probability of C on D_{i_0-1} being 1 that is what, it is plus another half times. Now this is probability of C not being 1.

Which will be 1 minus probability of C being 1 on y_{i_0} complement and remaining is z_{i_0+1} so on till $z_{i_m} = 1$. Let us further simplify this so first is so I will get half + p_{i_0} . So, I will rewrite it like this p_{i_0} and I have to subtract this was not p_{i_0-1} this is D_{i_0} . This is how we define the distribution so in D_i we take z_{i+1} this should be minus 1. And this is p_{i_0-1} plus I will put the half then I subtract half.

And what else do I have? I have minus half times this probability expression. What is this probability expression? Let me in fact club these two together so I will get plus the probability of $C(y_1, \dots, y_{i_0-1}, \overline{y_{i_0}}, z_{i_0+1}, \dots, z_{i_m})$ being equal to 1. So, now what is this thing inside the bracket? $p_{i_0} - 1$ plus this probability of C with y_{i_0} negated this is D_{i_0} because z_{i_0} is actually y_{i_0} . We are using up to y_0 prefix so that is by definition D_{i_0} .

If you look at this so this is the probability of C with y_1 to y_{i_0-1}, y_{i_0} and then z. What you see is probability of C being 1 plus a similar expression the only difference is first has y_0 the second has y_{i_0} complement. This probability sum will correspond to the distribution $i_0 - 1$. So, up to $i_0 - 1$ you are actually using the prefix and the remaining things you are taking random.

In other words what you can write is this is equal to $p_{i_0} + \text{half} - \text{half}$ and here you will have p_{i_0-1} times 2 because p_{i_0-1} is essentially prefix up to $i_0 - 1$ and the i_0^{th} is 1 with probability half you choose 0 or 1. So that is what the some of the probability is the average of that so which is why these 2. So, this gives you half + $p_{i_0} - p_{i_0-1}$. And that is a big advantage because you know that $p_{i_0} - p_{i_0-1}$ is a probability this probability difference is at least 0.1 by m.

You get this much of advantage so C' is actually better than just a random output it is actually making a significant improvement over half. That is why we can say that C' is a bit predictor.

This implies that C' is a decent bit predictor. The only issue is C' is using these random bits z_{i_0} to z_{i_m} so we have to make it somehow go away we have to make it deterministic.

What you can do is? You can actually just fix these bits z_{i_0} to z_{i_m} suitably so that the advantage is not lost. To make C' deterministic fix z_{i_0} to z_{i_m} suitably and get circuit C'' .

(Refer Slide Time: 25:03)

C'' : $\Pr_{y \in \text{NW}(U_e)} [C''(y_{i_1} \dots y_{i_{i_0-1}}) = y_{i_0}] \geq \frac{1}{2} + \frac{0.1}{m}$
 (Averaging argument)
 $\triangleright \text{size}(C'') \leq 2 \cdot \text{size}(C) \leq S/5$.
 - Plugging the defn. of NW_f^S , we get:
 $\triangleright \Pr_{z \in U_e} [C''(f(z_{I_1}), \dots, f(z_{I_{i_0-1}})) = f(z_{I_{i_0}})] \geq \frac{1}{2} + \frac{0.1}{m}$.
 Idea: Fix $z_{(I_j)_{j \in [i_0-1]}}$ s.t. the above prob. is retained.
 $\Rightarrow \forall j \in [i_0-1]$, z_{I_j} has all places fixed except $|I_j \cap I_{i_0}|$ many variables. ($\leq d$)
 $\Rightarrow f(z_{I_j})$ is d -variate.

So, C'' will continue to have the advantage in prediction where y is coming from the Nisan Wigderson stretch. Now there are no random bits be used because the remaining z is fixed to a good choice and so there will always be some good choice of fixing z again by the averaging argument. So, based on the averaging argument we have fixed z suitably so now they are gone and we have that C'' just takes $i_0 - 1$ prefix of NW image bit with a good advantage the advantage is 0.1 by m .

Now what is the size of C'' ? Size of C'' is the same as the size of C why is that? Because it is basically just using C it evaluates C and then either it will output z_{i_0} or the complement. It is just an R so the size is at most twice that of size of C kind of if then else which is then less than equal to S over 5 because C was size S by 10 . Plugging the definition of NW_f^S we get the following that the probability of C'' .

What are these y 's? y 's are just evaluations of f on the substring of Z using the design and using $i_0 - 1$ evaluations you are or C'' is predicting f on I_0^{th} substring within advantage of 0.1

by m that is what C'' is doing. Now the question is how do you deduce that f is easy from this? This seems to be taking values of f and giving another value of f . Somehow we have to remove f from the argument of C'' .

And the way we will do it is by fixing many of these Z . Let us fix Z other than i_0 on the RHS. Other than i_0 that part of Z you should fix and then in the left hand side hopefully you will remove the extra Z and you can then maybe say that f at Z_{i_0} is being evaluated on the left hand side that is the idea. Let us fix $Z_{[l]/I_0}$ sub string such that the above probability is retained.

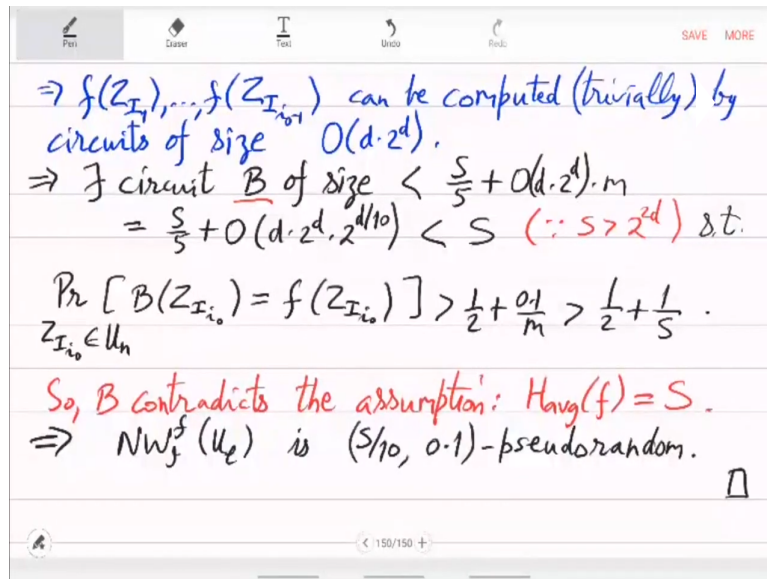
This means that for $j_{i_0-1}, 1$ to $i_0 - 1$, Z_{I_j} has all places fixed except I_j intersection everything other than I_0 is fixed. So, I_j intersection I_0 is what remains and what can you say about I_j intersection I_0 . So, I_j intersection I_0 we have to go back to the design we had a design that we were using in NW definition. In the design we said that intersections are smaller than D or at most D .

So that is what happens here so by the way this I have to correct this is not 0 this is at I_0 so I_j intersection I_0 is small that is less than equal to d . This means that $f(Z_{I_j})$ how many arguments are actually free at most d so, this is d variate. By just fixing in the left hand side these all these Z_{I_1} to $Z_{I_{i_0-1}}$ they will all become just d variate d is considered small. So, now we will do a trick of also eliminating them so this f is a hard function, remember that we started with an average case hard function.

In the left hand side our goal is to actually make this $f(Z_{I_1})$ extra easy to evaluate. So that we can claim that left hand side is easy right hand side is hard but still we are able to get an equality now that will happen because f on this Z_{I_j} after the fixing has become only d variate.

Since it is only on d arguments you can evaluate it in circuit size 2^d .

(Refer Slide Time: 33:30)



So, this means that $f(Z_{I_1})$ to $f(Z_{I_{i_0-1}})$ these functions since they are on few variables after the fixing they can be computed trivially by circuits of size 2^d . 2^d is the number of inputs that you are giving to this f and the size of the circuit will be d times 2^d basically there are these d variables and then they take 2^d to many values. You get this much sized Boolean circuit which means that there exists circuit B of size.

Now let us look at the whole thing on which was on the left hand side C'' prime-prime on the arguments f on I_1 and so on till f on I_{i_0-1} this whole thing what is the circuit size of this. So, this we will see as a circuit B and you can estimate the size as C'' has size S by 5 then each evaluation of f we have written trivially as d times 2^d and i_0 is at most m . So that is the size estimate which is equal to $S/5$ plus what is m ? m comes from the design.

If you go back to the design you will see that m is around $2^{d/10}$ that is the estimate and you can see that this is all less than S this again will come from the value that S in lemma 2 we assume that the average case hardness of f is more than 2^{2d} that is what figure we are assumed. Because of that so clearly 2^d is much more than this overhead that you are getting of 2^d times $2^{d/10}$.

And even if you add $S/5$ to that you will remain under S . Point being that B is a circuit of size quite small. In that sense now you are getting that f is not hard at all and what is the how good is B computing f so the probability of B on $Z_{I_{i_0}}$ being equal to f at $Z_{I_{i_0}}$ on the inputs $Z_{I_{i_0}}$.

How big this is set I? This is an n sized subset. So, on this domain of U_n B evaluate safe with advantage 0.1 by m which is more than $\frac{1}{2} + 1$ by S .

Because S is greater than m , m is around $2^{d/10}$. S is more than 2^d . The advantage is more than $1/S$. It seems that f is being evaluated by B pretty well and that is a contradiction because we had assumed that f is average case hard. So, B contradicts the assumption that average case hardness of f is S because B is the circuit of size smaller than S and it is able to compute f on more than half sufficiently away from half many inputs.

So, what does this give you in the end? This contradiction ultimately implies that what we had assumed about NW 's output being not pseudorandom it actually is. So, we deduce that NW_1^f on U_1 is $(S/10, 0.1)$ pseudorandom. This finishes the proof of lemma 2. Lemma 1 showed that there is a design easy to compute lemma 2 showed that using the design.

And the average case hard function you have a pseudorandom distribution given by NW map. Now using these 2 lemmas it is actually quite easy to prove the main theorem of Nisan Wigderson you just have to do some parameter chasing.

(Refer Slide Time: 40:46)

Proof (of NW-Theorem): Let $f \in E = Dtime(2^{O(n)})$ &
 $Hard(f) \geq S(n)$.

- Define $S(\ell)$ -prg G : On input $z \in \{0,1\}^\ell$:
 - 1) Pick n s.t. $\frac{100n^2}{\ell S(n)} < n \leq \frac{100(n+1)^2}{\ell S(n+1)} \leq \frac{200n^2}{\ell S(n)}$
 - 2) Set $d := \ell S(n)/10$. (Lemma 1)
 - 3) Compute an (ℓ, n, d) -design $J := \{f_1, \dots, f_n\}$; $m := 2^{d/10}$.
 - 4) Output $NW_J^f(z)$. computing f
- This takes time: $2^{O(\ell)} + 2^{O(n)} \cdot m \leq 2^{O(\ell)}$.
- Since $Hard(f) \geq S(n) = 2^{10d}$ by Lemma-2 we get:
 $NW_J^f(U_\ell)$ is $(S(n)/10, 0.1)$ -pseudorandom.
- \triangleright The stretch is $m = 2^{d/10} = S(n)^{0.01} =: S'(\ell)$

Let us do that so let f be in E which we call is to 2^n time and average case hard let us see the hardness at least a $S(n)$ function. Define a stretch function $S'(l)$ prg which it will stretch l bits to m bits using the design basically this NW map so on input $z \in \{0, 1\}^l$ what you do is pick n such that basically want n I would say something like square root l to have a design you need l and d of certain type.

So, you want l to be slightly quadratic around quadratic of n so pick an n like that. Do not worry about these parameters so take it to be this so l you can think of it as n^2 it is quadratically related. Now there is a design (l, n, d) -design use that. So, compute and (l, n, d) -design, you can take to be $2^{d/10}$. This (l, n, d) -design is just a set of or family of sets these are $2^{d/10}$.

Their mutual intersections are smaller than at most d each of them is an n subset of l to l . Remember that is the design you can compute it in time 2^d . It is very explicit and using this you can stretch Z so output this NW stretch this will stretch l to m and in how much time this takes time 2^l plus basically just we have already seen the design. That is it just that you have to also look at what NW does NW is it is just evaluating f on n bits.

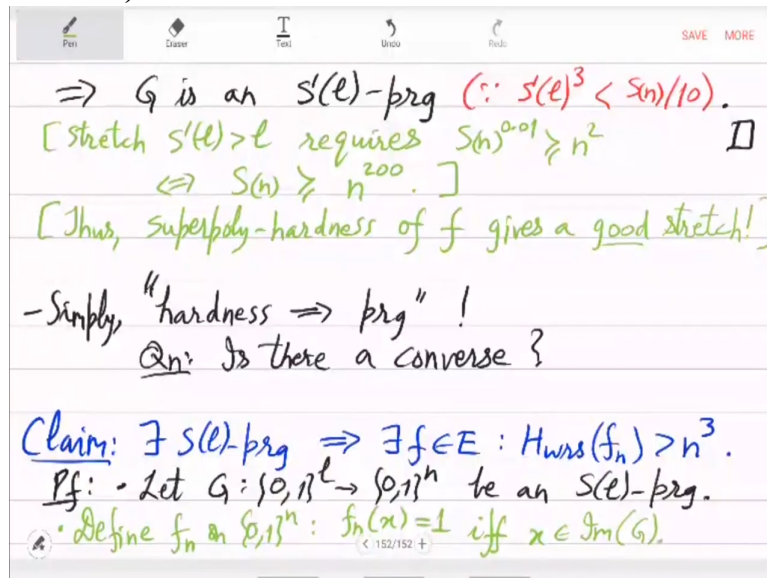
That you can do in 2^n time and this you have to do m times so, overall this is still 2^l . Let me just mention that this is for computing f by the truth table you can compute in 2^n time so you are using that and 2^l is for the design. So, n is obviously smaller than l and m is just something like 2^d so you still get 2^l it is fast.

It is we can see it is explicit now since the average case hardness of f is at least $S(n)$. And $S(n)$ by choice it is 2^{10d} . Then we can use lemma 2, lemma 2 said that average case hard function and design you can use also you can note that the parameters I have chosen (l, n, d) -design exists by lemma 1. So, you just wanted l to be more than $10 n^2/d$ and that is happening here l is more than $10 n^2/d$.

This is by lemma 2 the design is by lemma 1. Design exists and then NW is computed in this much time and using average case hardness this assumption you get that $NW_l^f(U_l)$ is $(S(n)/10, 0.1)$ -pseudorandom. We have a pseudorandom distribution and we have a prg which is this NW map it is stretching l to m . How much is the stretch? The stretch is $2^{d/10}$ which is again is $S(n)^{1/100}$.

For average case hardness $S(n)$ the stretch that your prg is giving is $S(n)^{0.01}$. So, stretch is not $S(n)$ but it is slightly smaller and this function we can call S' . Remember l is like quadratic in n . So, l is being stretched to $S'(l)$ which is actually $S(n)^{0.01}$. $S(n)$ has to be sufficiently large for this to give something interesting.

(Refer Slide Time: 50:00)



$\Rightarrow G$ is an $S(l)$ -prg ($\because S(l)^3 < S(n)/10$).
 [stretch $S(l) > l$ requires $S(n)^{0.01} > n^2$ \square
 $\Leftrightarrow S(n) > n^{200}$.]
 [Thus, superpoly-hardness of f gives a good stretch!]
 - Simply, "hardness \Rightarrow prg"!
 Qn: Is there a converse?
 Claim: $\exists S(l)$ -prg $\Rightarrow \exists f \in E : H_{wrs}(f_n) > n^3$.
 Pf: - Let $G: \{0,1\}^l \rightarrow \{0,1\}^n$ be an $S(l)$ -prg.
 Define f_n on $\{0,1\}^n$: $f_n(x) = 1$ iff $x \in \text{Im}(G)$.

So, this implies that G is an $S(l)$ prg note that in the definition of prg we had this $S(l)^3$ circuit. You should observe that $S(l)^3$ is smaller than $S(n)/10$ and since we have shown that NW images $S(n)/10$ you do get that it is an $S(l)$ prg. That finishes the proof this stretch is interesting only when $S(l)$ is $S(n)^{0.01}$ and l is around n^2 so you want this to be larger than n square.

You want $S(n)$ to be sufficiently bigger than n^{200} . So, basically the point is that average case hardness should be something like n^{200} or more for this theorem to be interesting because in that case only the stretch is actually bigger than l that it will make sense only in that case and surely thus super poly hardness of f gives a good stretch. We have proved this very important theorem that average case hardness sufficiently high gives you a prg.

And it will be a good stretch if you are working with a super poly hard Boolean function. We can say that hardness implies prg. Now is there a converse to this hardness in fact this average case hardness implies explicit prg existence thus the explicit prg imply or this prg implies hardness for an explicit function that also is true not a very stunning connection but just to understand the definitions you can do as an exercise.

You can show that if you have an $S(l)$ prg then there is a Boolean function in E such that the worst case hardness is cubic. You will not get a very stunning connection but it is kind of this super cubic hardness of an explicit function is what you get this really follows from the

definition. Let G be an $S(l)$ prg so n is $S(l)$ and that is bigger than l . Now what do you think is this function $f(n)$ that is a candidate for hardness it has to be related to G to the prg.

Why not just look at the image of G and those strings you call the S strings in the space and the remaining strings you call now that defines a Boolean function $f(n)$. Define $f(n)$ on $\{0,1\}^n$, $f(n)$ is 1 if and only if x is in the image of G . Now since G is a prg it is computable in E so $f(n)$ is also computable in E .

(Refer Slide Time: 56:53)

$\Rightarrow f \in E.$
 • Let C_n be the smallest circuit computing f_n .
 $\triangleright \Pr[C_n(G(U_l)) = 1] = 1.$
 $\triangleright \Pr[C_n(U_n) = 1] \leq 2^l / 2^n \leq 1/2$
 $\Rightarrow C_n$ distinguishes $G(U_l)$ from U_n .
 $\Rightarrow \text{size}(C_n) > S(l)^3 = n^3. \quad \square$

Boolean function computable in E let me call this f so let C_n be the smallest circuit computing $f(n)$ so what is the circuit size? What is the size of C_n ? That is the question. So, the thing is that what you can show is that immediately you get that C_n on the image of G it has to be 1 all the time and on the other hand what is C_n on U_n ? How frequently is it 1? Now recall that the image of G has only 2^l strings.

You can estimate this as 2^l by 2^n which is definitely smaller than half. So, C_n and U_n is less than half times many times 1 while C_n on the image of G is always 1. So, C_n is able to distinguish the two distributions C_n distinguishes $G(U_l)$ well from the uniform distribution U_n very well. So, which means that by the definition of prg that size of C_n has to be more than the stretch cube and what is the stretch $S(l)$ is m .

What we have shown is that the complexity of this Boolean function f is at least cubic that is the lower bound. So, this is not highly interesting just follows from the definition that prg implies some hardness but hardness connection to prg is much more impressive.