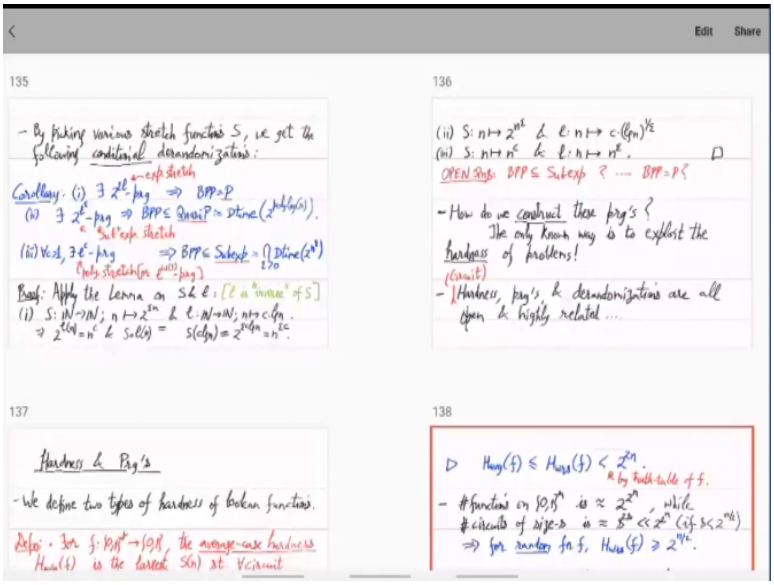
Randomized Methods in Complexity
Prof. Nitin Saxena
Department of Computer Science and Engineering
Indian Institute of Technology – Kanpur

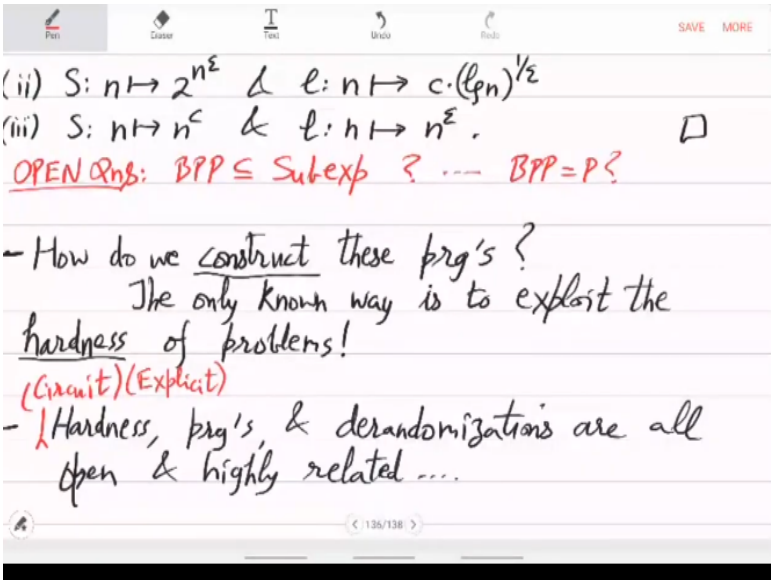
Lecture – 18
Hardness Vs Randomness

(Refer Slide Time: 00:15)



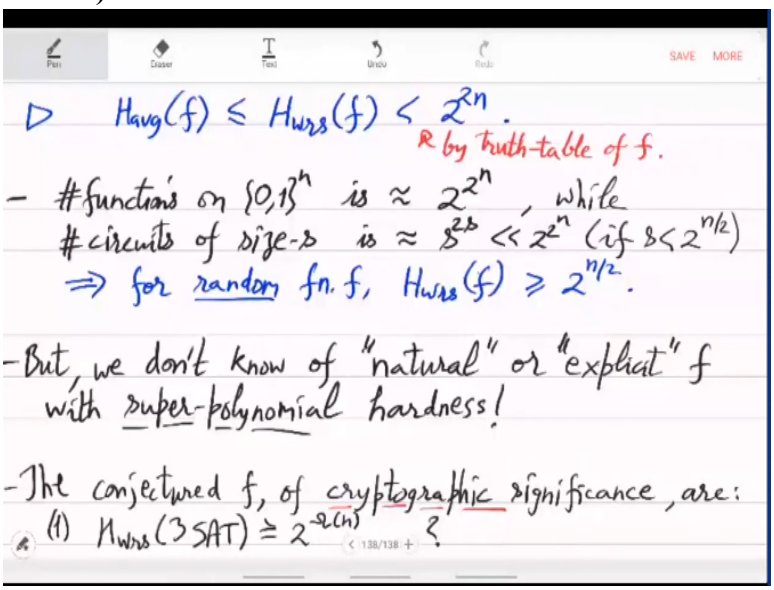
So, in the last class we defined prg then hardness of 2 types average case, worst case hardness of any Boolean function. This hardness, remember we will be in terms of circuit hardness. So, that is harder than actually the concept of hardness is harder than about algorithms, this is about Boolean circuits. Now our goal is and it will remain this goal for the remaining course to connect circuit hardness with prg's and we have already shown that once you have a prg you have some kind of de randomization.

(Refer Slide Time: 01:02)



So, this hardness will be for circuit hardness and moreover to get practical prg we will need actually explicit function which is hard. So, it will be circuit and will be explicit. So, remember these 2 terms explicit means that your function is computable in a decent amount of time on during machines there should be an algorithm. It should not be just an arbitrary Boolean function but it should be somehow computable usually will say that it is computable in E 2^{2n} time.

(Refer Slide Time: 01:48)

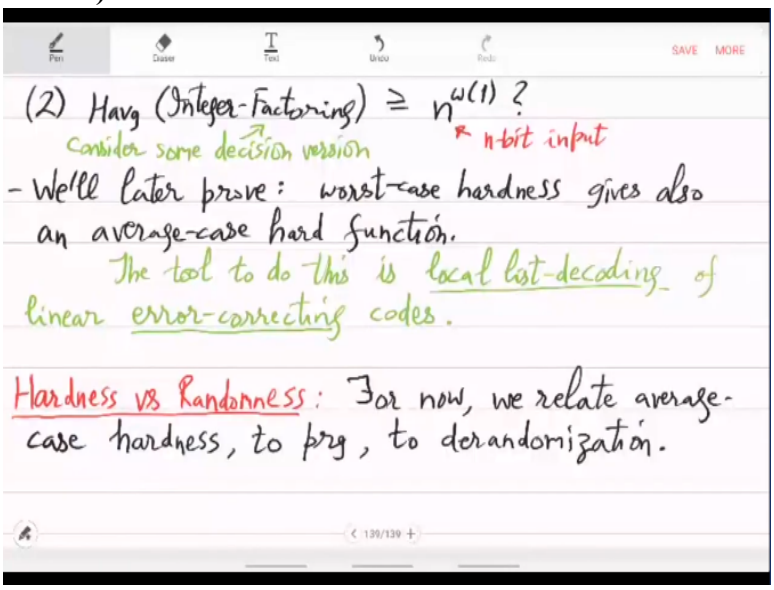


Then we had defined the average case, worst case hardness and currently it is an open question to find natural or explicit Boolean functions which are super polynomial hardness worst case, average case everything is open. Although we know that most of the Boolean functions are hard,

we know that if you just pick some arbitrary Boolean function then on n inputs, it will require circuits of the size $2^{n/2}$, but then it is not explicit.

So, can you think of examples explicit functions conjecture to be hard, so the conjectured f of cryptographic significance are so there are 2 explicit functions you can think of one is the problem of SAT, so what will be the circuit size required for SAT and remember that SAT is solvable in 2^{2n} time. So, it is explicit enough, but is it hard enough. So, it is conjecture that $H_{wrs}(3SAT) = 2^{\Omega(n)}$. It is believed to be exponentially hard in the worst case may not be in the average case, but worst case it is believed to be very hard. And second is the problem of integer factoring.

(Refer Slide Time: 04:00)



So although it is a functional problem, you can make it a decision problem by saying that given a number you want a prime factor of it or the smallest prime factor of it and you want some bit of that prime factor, see the middle bit. So that way you can make it a decision better than that what is n? n is the input size it remember? So for an n bit number, the magnitude is to 2^n and better than that algorithms are known actually.

So it is not believe it is not exponentially hard, but at least super poly hard we believe in average case. So, consider decision version, so later on in the course, we will show we later proved that worst case hardness function implies an average case hard function. So, a priori it may not seem

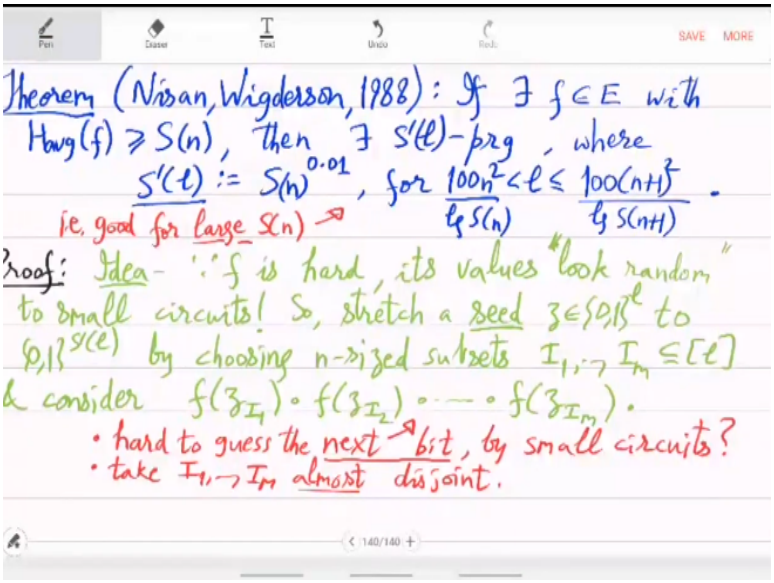
possible, you may think that worst case hardness only means that the problem cannot be solved on all the inputs whereas average case means that on average the problem is unsolvable.

That is that seems much stronger, but we will actually come up with a very clever technique which can distribute this worst case hardness across inputs, thus making the function average case hard. So, that will be the new tool we will study towards the end of the course. So, the tool to do this is called local list decoding of linear error correcting codes, so not just decoding but list decoding and on top of that local list decoding. So, they will achieve this in many steps that is what are will be busy towards the end of the course.

So, for now, what we will do? Assuming average case hardness we will construct prg's, we have already seen that prg's give you the randomization. Now, we will go one step back and show that actually average case hardness also gives prg's. So, in turn and then if when you combine it with worst case hardness with this result to be proved towards the end of the course, you will get that worst case hardness implies complete de randomization or some de randomization. So, this topic is called hardness versus randomness.

For now, we relate average case hardness to de randomization which is to prg and then to de randomization. So, what this result philosophy says that both hardness and randomness cannot coexist in nature, if hardness is there, then randomness is not really there. And we always meet in terms of efficient algorithms any randomized efficient algorithm can be converted into deterministic, efficient algorithm and there are also connections between the other ways. So, if there is a prg then there is also hardness. So, in that things these 2 are opposites this is why hardness versus randomness.

(Refer Slide Time: 10:10)



So what is this mathematically? So let us see the theorem,(Nisan,Wigderson,1988): If $\exists\, f \epsilon E \; with \; H_{avg}(f) \geq S(n), then \exists\, S'(l) - prg \;\; where \; S'(l) = S(n)^{0.01}, \; for \; \frac{100n^2}{log\, S(n)} < l \leq \frac{100(n+1)^2}{log\, S(n+1)}$ so this is the precise definition of a S'. So, the stretch that we are looking at l basically, the idea is that we will use this f whose input was n bits to stretch l bits and l will be around n square and this l will be stretched to $S'(l)$ which is actually $S(n)^{0.01}$. So, this will be interesting already if $S(n)$ was sufficiently large polynomial in n. $S(n)$ was n^{1000} then you can see that this will be an interesting stretch, so this is good for larger $S(n)$.

So how do you prove such a result? If somebody gives you a hard function, how do you stretch l bits? So, that in the image the distribution that you are getting looks random to small circuits, how do you do that? So, remember the definition of prg in fact, remember the definition of pseudo random distributions. So, it should look random to small circuits. So the way you can use f is that the image of f will look will not be computable by small circuits because of the average case hardness.

So use that fact so that f output looks random, quote unquote random to small circuits, so use that fact.

Proof :

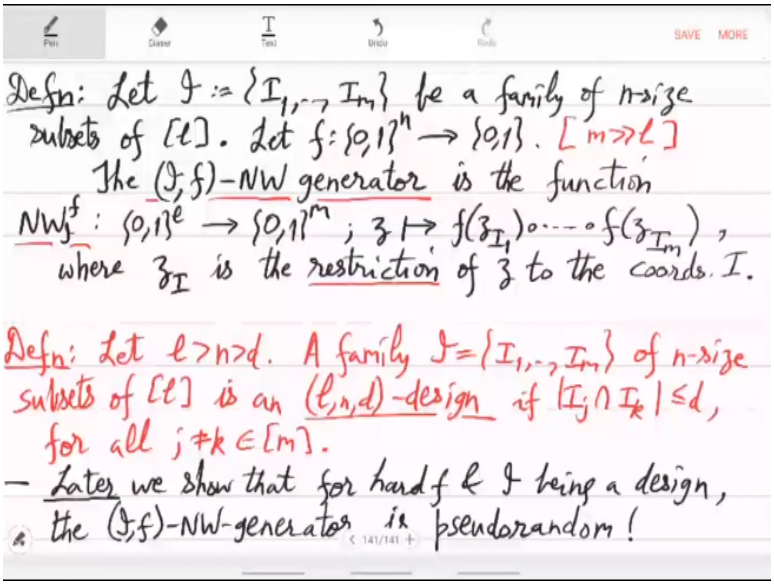
since f is hard it is values look random to small circuits, So stretch a seed $z \in \{0,1\}^{l}$ to the $\{0,1\}^{S'(l)}$, by choosing n sized subsets $I_1 \ldots I_m \subseteq [l]$ and consider $f(z_{I_1}).f(z_{I_2}) \ldots\ldots f(z_{I_m})$

Now, this is how you can stretch l bits to m bits by evaluating f many times on many subsets, or many substrings of z. Note that this m bit string has a good chance of looking random to small circuits because the bits seem hard to predict from what comes before. So hard to guess the next bit by small circuits and because of this belief that the bits are unpredictable in this string by looking at the prefix we could say that or we could expect the string to be a pseudo random distribution.

And we believe that the next bit is unpredictable because, naively it would seem that to predict the next bit, you have to f inverse and you have to work with f inverse and then you have to apply f. So, all those things, we do not expect small circuits to be able to do that is the rough idea. And now we will build on this mathematically. So take $I_1 \ldots I_m$ almost disjoint. So, if you take them disjoint there it becomes even better because to predict the next bit you actually have to evaluate f, the prior information the prefix does not give you anything does not tell you anything what the next bit will be.

So you actually have to evaluate f but evaluation of f is considered hard for small circuits so that is the plan. So, let us formalize this construction in 2 steps so in the first step, what I will do is define this ambit string.

(Refer Slide Time: 19:27)



Definition: let I :$\{I_1 \dots I_m\}$be a family of n -size subsets of $[l]$, let $f: \{0,1\}^n \to \{0,1\}$ be a function Boolean function so, from set family and this function we can define that string and it is called (I, f) Nisan Wigdersan's generator, so Nisan Wigdersan's are the authors of that theorem.The (I,F) NW generator is the function $NW_i^f: \{0,1\}^l \to \{0,1\}^m$; $z \to \quad f(z_{I_1}) \circ \dots\dots \circ f(z_{I_m})$, l bits will be stretched to m bits think of m is much bigger than l. Where z_I is the restriction of z to the coordinates I. Previously also when I said z_{I_1}, I meant these positions of set. So, this NW generator is formalized and we have to study and then we have to prove under what conditions is this a pseudo random distribution.

So that so we will show that actually when f is hard and these $I_1 \dots I_m$ are disjoint almost disjoint then this is a pseudo random string, but before that let me define disjointness what do we mean by that

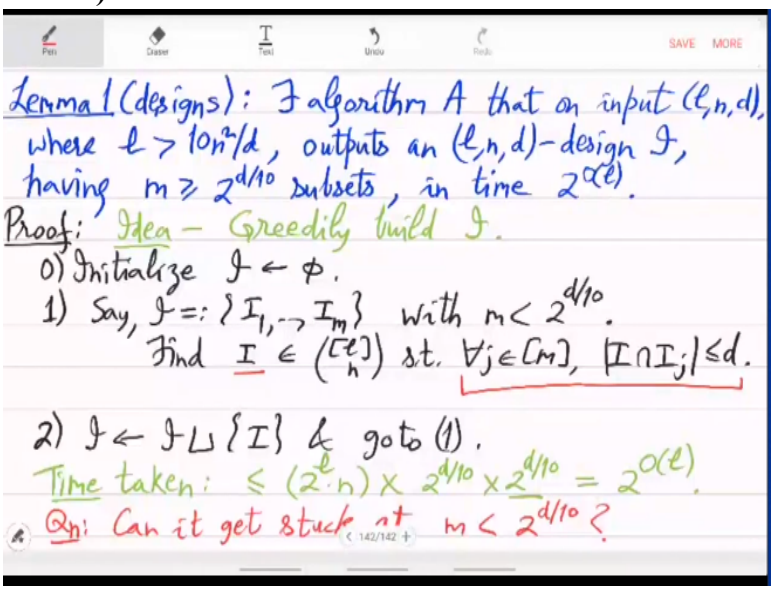
Defn :

$Let\ l > n > d.\ A\ family\ I = \{I_1 \dots I_m\}\ of\ n\ -\ sized\ subsets\ of\ [l] is\ an\ (l, n, d)\ -\ design\ if$ $|I_j \cap I_k| \leq d\ for\ all\ j \neq k \epsilon [m]$So, we have defined this notion of design which is really saying that $\{I_1 \dots I_m\}$ are mutually almost disjoint we will take d to be smaller, much smaller than l.

Now, this disambiguation generator the string will show that this is you do not random if f is hard, average case hard and the family is a design. So, later we show that for hard f and I being a design the (I , f)-Nisan Wigdersan's generator is pseudo random, so hardness and being a design suffices that is how we characterize the randomness or pseudo randomness of this Nisan Wigdersan's generator.

But before that we have to show some things first we have to show does the design exist, we do not know whether average is hot functions exist, but we can at least prove that our design exists.

(Refer Slide Time: 25:31)



So let us first do that and not only does this design exist, we can actually construct it efficiently. So,

Lemma 1:

∃ an algorithm A that on input (l, n, d) where $l > 10n^2/d$,outputs an (l, n, d) design I having $m \geq 2^{d/10}$ *subsets in time* $2^{O(l)}$

So, there are 2^l subsets of l to l and essentially that is the time to construct this design, it is kind of enumerating going over all the subsets, but it is not trivial. In fact, you do not even see why it exists. Because you want so many subsets with mutual interest section smaller than d, so why

such a thing should exists and the size is also only n of every subset. So let us give this algorithm what is the idea of this? So idea is a greedy approach.

Proof: Idea:

Find a subset I_1 I mean just pick any subset I_1 and then pick it I_2 so that it is sufficiently disjoint from I_1 and thenI_3 pick it so that it is sufficiently disjoint from the previous 2 and so on, so greedily build, the family I .So initialize I← ϕ.

1)Say I=:$\{I_1 \ldots I_m\}$ with m<$2^{d/10}$

So, suppose at some point in the algorithm you have already found m subsets, they are n size, their mutual intersections are less than equal to d. But the subsets are not enough, they are smaller than $2^{d/10}$ n numbers. So, you have to pick one more, so how do you pick one more? Over all the remaining subsets. So, find $I \epsilon \frac{[l]}{n}\ \forall j \epsilon [m]\ |I \cap I_j| \leq d$,

Now, this I may not exist, we will analyse that situation later, but suppose you find it then you just include it in the family

2)I←I⊔$\{I\}$& go to 1. So, simple step by step or a greedy approach the only issue is that in step 1 at some point it may get stuck. So, you have to show that it actually goes all the way to $2^{d/10}$ iterations, time that it takes is just a simple computation you are going over all the subsets in 1 iteration, so that is $2l$ and subsets are of size n.

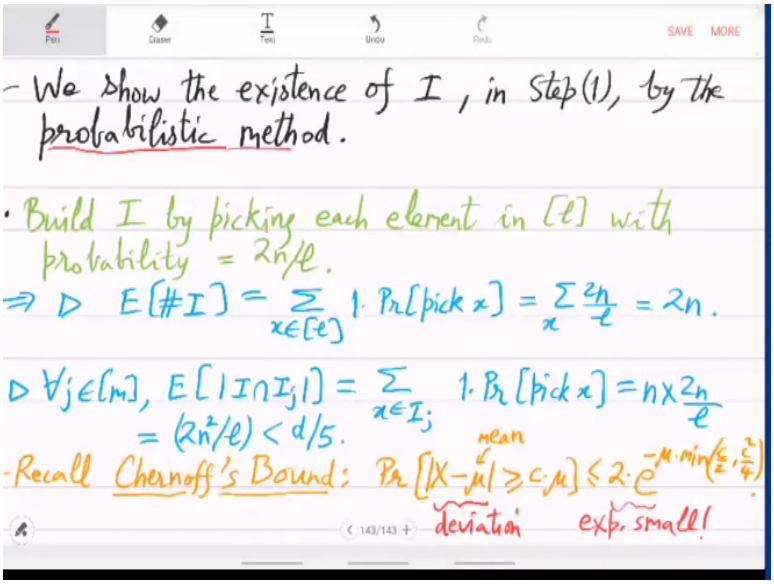
So, this much time in a single iteration, and the number of iterations that you are doing is $2^{d/10}$. That is the number of iterations and this m that you have collected you will be looking at the intersection of I with them. So, that is another 2 ways to do it in every step,

$Time\ taken: \leq (2^l . n) \times 2^{d/10} \times 2^{d/10} = 2^{O(l)}$

so that is in all $2^{O(l)}$ so time is within bounds as claimed, can it get stuck at m < $2^{d/10}$. So, we will show that does not happen it is a surprising fact that step 1 will always find I.

So, we have to show that in this algorithm when in step 1 we are looking for I that is almost disjoint from each of these I j. In particular, the intersection should be less than equal to d this always exists as long as m is small, smaller than $2^{d/10}$

(Refer Slide Time: 34:09)



So, we show it will not get stuck and this we show the existence of I, so in the in step 1 by the probabilistic method. So what this means we will now show that in step 1, when you are searching for it, if you do the searching random way that will surprise which in particular will mean that if you do a group forces you will find an I. Because probabilistically it exists that is the reason probabilistic analysis will be easy to do.

So, instead of showing it by some other means, we will prefer the probabilistic method, you will see that it is very easy to show. So, build I by picking each element in $[l]$ with probability $=2n/l$, there are l elements if you did this uniformly, you would have given probability $1/l$. But that will in the expectation it will not give you too many elements since you want many elements, you want I you give each element more probability to $2n/l$ probability.

And you pick which will ultimately mean that expectation is you will be picking many elements that is the probabilistic process. Now, let us analyze this process, so what you can write is that

$E[\#I] = \sum_{x\epsilon[l]} 1.Pr[pick\ x] = \sum_x \frac{2n}{l} = 2n.$ So, in expectation, you are picking 2n elements, this is just by the definition of expectation. Moreover,

$\forall j\epsilon[m], E[|I \cap I_j|] = \sum_{x\epsilon I_j} 1.Pr[pick\ x] = n = n \times 2n/l = (2n^2/l) < d/5$

So, this process is doing good that is what you will learn. So, what remains to be done is we have to compute the probability or estimate the probability for this size of I being n it is a probabilistic process. So, there is a chance that the size of I is smaller than n, if it is bigger than n it is because we can just drop elements more than n but if it is smaller than n, then it is not a good, it would not work for us because we wanted I to p n size subset.

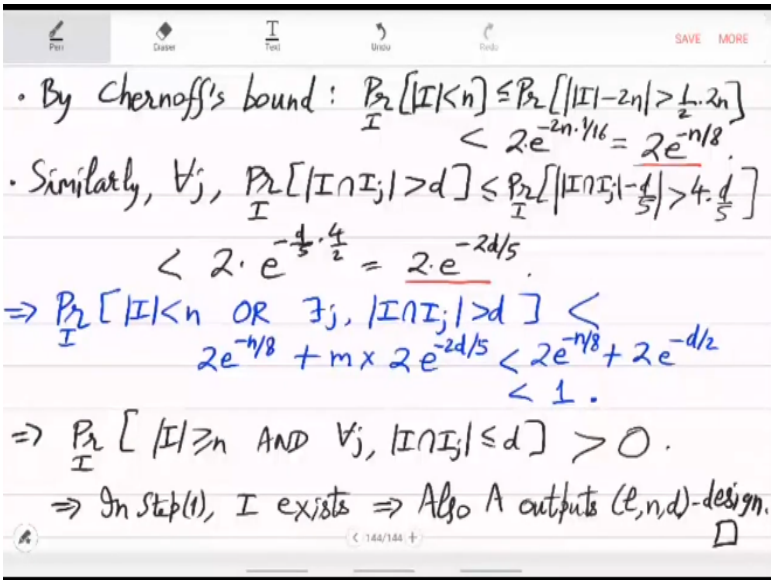
And similarly, what if the intersection of I with I_j is more than d again, this process is a probabilistic process. So it may give you I intersection I_j more than d, so we have to estimate those probabilities and that we will do by Chernoff's Bound.

Chernoff's Bound :$Pr[|X - \mu| \geq c.\mu]$so this is the mean, so this probability you expect to be small.

But Chernoff's Bound will give you very specific estimate it will say that $Pr[|X - \mu| \geq c.\mu] \leq 2.e^{-\mu\, min(\frac{c}{2}, \frac{c^2}{4})}$so,don't worry too much about this main $(\frac{c}{2}, \frac{c^2}{4})$. But think of this as $e^{-\mu c}$, so what this is saying is that the deviation away from the mean by a multiple of c is exponentially falling, it is $e^{-\mu c}$. So, this probability is exponentially small that is what Chernoff's Bound says and you can also prove Chernoff's Bound maybe we will give this as an exercise.

This is an important and very useful estimate in probability and now we will use it so, now we will relate, so, remember mean and expectation are the same. So, we have computed the expectation and now we will compute the deviation.

(Refer Slide Time: 42:45)



So now, *by Chernoff's Bound* :$Pr_I[|I| < n] \le Pr[||I| - 2n| > \frac{1}{2}.2n]$.

$$< 2.e^{-2n.1/16} = 2e^{-n/8}$$

, so point being that this is exponentially small in terms of n this is a very small probability that size of I will be smaller than n it will be n or more which is good enough for us we wanted n subset. Similarly, $\forall j, Pr_I[|I \cap I_j| > d] \le$ $Pr_I[||I \cap I_j| - d/5| > 4.d/5] < 2.e^{-\frac{d}{5}.\frac{4}{2}} = 2.e^{-2d/5}$ So, you get these 2 probabilities as you can see both are exponentially small. So, now what you get is the $Pr_I[|I| < n \ OR \ \exists j \ |I \cap I_j| > d] < 2e^{-n/8} + m \times 2e^{-2d/5} < 2e^{-n/8} + 2e^{-d/2} < 1$ for large enough and indeed this sum is actually very small much smaller than 1.

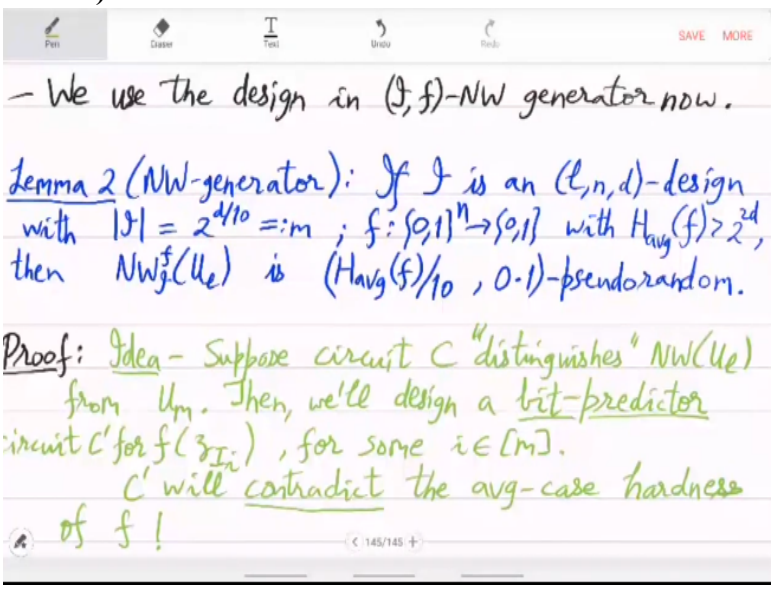
$\Rightarrow Pr_I[|I| \ge n \ AND \ \forall j \ |I \cap I_j| \le d] > 0$

which means that in step (1) I exists same applies that the algorithm constructs I. So, this algorithm which is actually deterministic algorithm it will all the step 1 will keep finding I and ultimately it will give you this family which is a(l, n, d) design 2^l time.

So, once you know that design is not only there but also constructible this way what you can do is use the hard function f so, we were here we had defined this Nisan Wigdersan's generator (I, f)NW generator. So we will now use $I_1 \ldots I_m$ from the design and f this average case hard function

and we will study the string of m bits the stretch from l to m will show that that is a pseudo random distribution.

(Refer Slide Time: 50:05)



We use the design in(I, f)NW generator so, we will get this

Lemma 2(NW generator): If I is an (l,n,d) design with $|I| = 2^{d/10} =: m;\ f: \{0,1\}^n$ with $H_{avg}(f) > 2^{2d}$ *then* $NW_I^f\ (U_l)$ *is* $(H_{avg}(f)/10, 0.1)$- pseudo random.

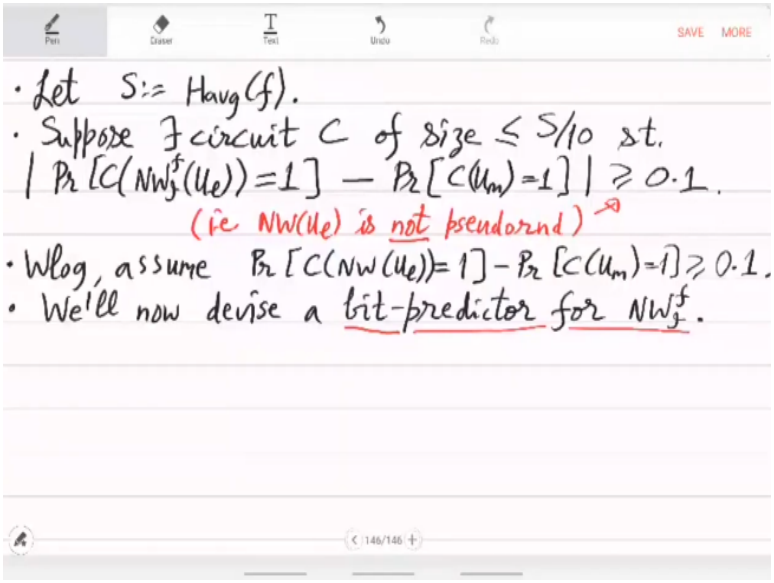
So, 2 circuits of size this parameter H the hardness of f /10 and we cube it so, the cube of this 2 circuits of that much size Boolean circuits they will not be able to distinguish this Nisan Wigdersan's output with the uniform distribution better than the probability distribution will be actually worse than 0.1. So, you can say in simple words that Nisan Wigdersan's output is pseudo random or looks random to small circuits.

So, why is that? How do you show this to suppose not it is a proof by contradiction,

Proof : *Idea*:suppose there is a circuit of small size C that is able to distinguish $NW(U_l)$ from U_m.Then you will 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. so, the overall sequence of arguments will be like this that if there is a distinguisher for $NW(U_l)$ versus U_m then that distinguisher will actually give you a bit predictor

and that bit predictor will have a small size and that will be in contradiction with the hardness of f. So, let us start this we will finish it next time, but let us start some implementation of this idea.

(Refer Slide Time: 56:42)



So, let S = $H_{avg}(f)$

Suppose $\exists\, circuit\ C\ of\ size \leq S/10\ st\ Pr[C\ (NW_I^f\ (U_l)) = 1] - Pr[C(U_m) = 1]$. So, remember we will try to C is we are claiming that or we are assuming that c can distinguish between these 2 distributions, stretch of U_l and U_m.

So, suppose this probability difference is more than 0.1 at least 0.1, so this is that is $NW(U_l)$ is not pseudo random. Then there is a distinguisher circuit C and we can assume that the difference is actually positive and at least 0.1 it is magnitude. So, you have to consider 2 cases it is positive more than 0.1 or it is negative and less than -0.1. So,

let us assume this case $Pr\ [C(NW(U_l)) = 1] - Pr[C(U_m) = 1] \geq 0.1$ We will now devise a bit predictor as promised for NW_I^f . So, this we will finish in the next class.