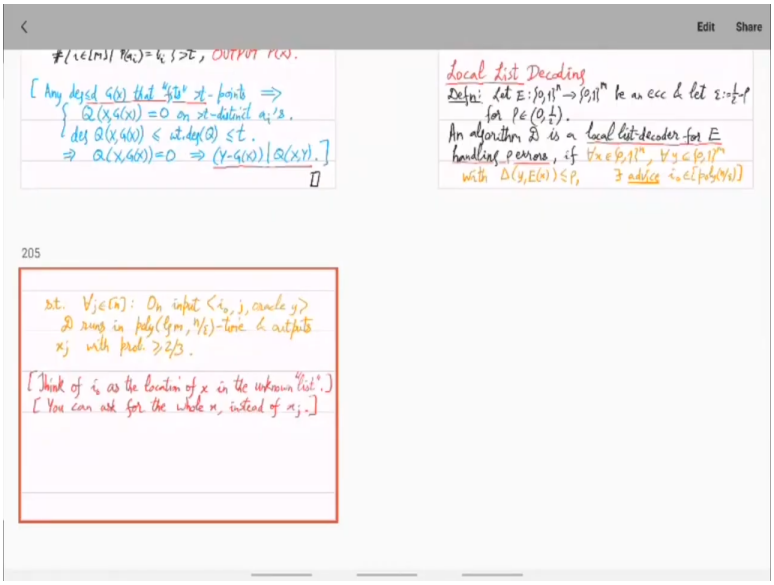
Randomized Methods in Complexity
Prof. Nitin Saxena
Department of Computer Science and Engineering
Indian Institute of Technology - Kanpur

Lecture - 26
Local List Decoding

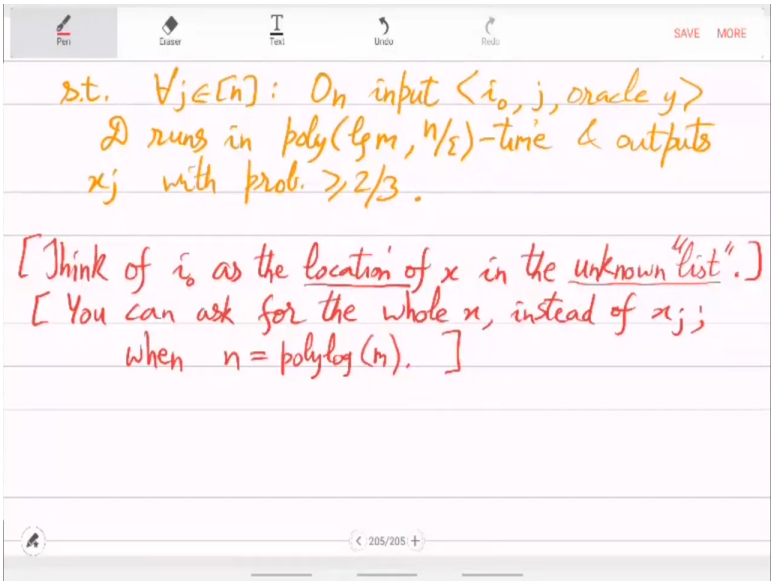
(Refer Slide Time: 00:13)



In the last class we started local list decoding we have done unique decoding we have done local decoding we also did list decoding for Reed Solomon. Now we will do local list decoding and by this we mean the concept of finding again jth bit of x message x given a corrupted codeword y where the errors are less than ρ but now the thing is that ρ can be very close to half really be close to half in this binary code which means that there cannot be unique decoding.

So, there may be many access and the idea is that given an advice what x through this string i_0 or through this number i_0 given this advice that I want or the user wants i_0th x j can be found with high probability and in very, very few queries it should be a poly$(logm, n/\varepsilon)$time where epsilon is how close you are to half in terms of errors.

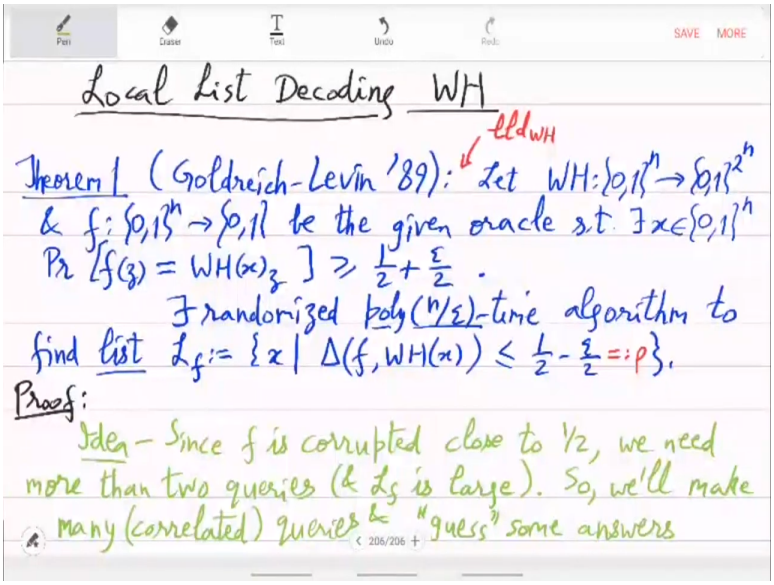
(Refer Slide Time: 01:32)



So, what I said was that you can also ask for the so you can think of i_0 as the location of x in the unknown list there is a list of x's and i_0 you can think of as the location and you can also ask for not just x_j but the whole string x assuming that it is not very long so n was the length of x. So, when n is in terms of time which is poly log m so when n = polylog (m) so which happens for example in the Walsh Hadamard case but it may not happen in Reed Solomon, Reed Muller case.

So, in case the blow up that the error correcting code is doing it is too much then you only ask for a few bits of x otherwise you can ask for the whole x because it makes sense to output everything why just jth bit. So, advice is the only new thing which has been added here. Let us now start this for the maps that we have seen.

(Refer Slide Time: 02:59)



So, let us start with **local list decoding Walsh Hadamard** I will not go into the full analysis because our time is up today is the last class. So, I will only give you the core ideas of local list decoding and then in the end we will finish the proof of average case hardness from worst case hardness. So, the first local list decoder is for WH given by Goldreich - Levin :

$Let\ WH: \{0,1\}^n \to \{0,1\}^{2^n}\ \&\ f: \{0,1\}^n \to \{0,1\}\ be\ the\ given\ oracle\ st\ \exists\ x \epsilon \{0,1\}^n$

$Pr[f(z) = WH(x)_z] \geq \frac{1}{2} + \frac{\varepsilon}{2}$

$\exists\ randomized\ poly(n/\varepsilon) - time\ algorithm\ to\ find\ list\ L_f := \{x | \Delta(f, WH(x)) \leq \frac{1}{2} - \frac{\varepsilon}{2} =: \rho$

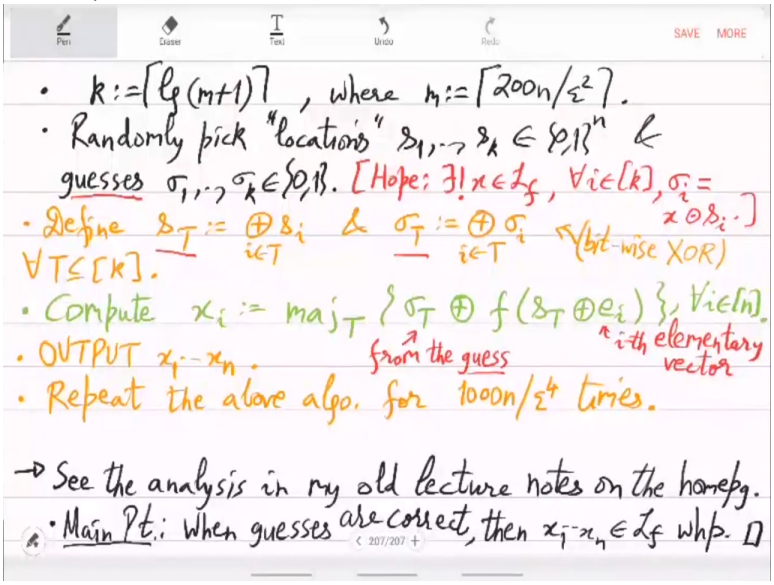
So, we can call this the local list decoder for Walsh Hadamard. So, the errors that it can tolerate is just below half anything below half it can tolerate what is the idea? The idea of local decoder was randomly choose a location and then to find the jth bit that location plus the elementary vector e_j. So, it query 2 locations and from that it constructed x_j since you want all the bits many, many locations will be queried.

And actually the most serious problem is that the corruption is almost half of the location. So, you cannot just randomly pick a place and hope to get x_j with probability more than 2/3 you wanted probability more than 2/3. So that will not be possible now because already the error is close to half in y or in f given by f. So, we will actually make more queries and from many, many queries we will try to deduce x_j and then repeatedly by repeating that will find x.

Proof: since f is corrupted close to 1/2 we need more than 2 queries and x is not unique and L_f is not a singleton L_f is long. So, there are many x's basically what we will do is we will make many correlated queries and we will get the answers and guess some answers. So, this guess will be basically to using the advice from advice. So, the advice let us ignore that for now in the theorem statement there is no advice.

Because here actually we are just we want to output all the x's but according to the particular x you want to output it will be obtained by a guess you can think of it as the advice that is the plan we plan so let us implement this carefully.

(Refer Slide Time: 11:30)



Let us define k = log (m + 1) where m := $[200n/\varepsilon^2]$. So, randomly pick 'location' $s_1 \ldots s_k \in \{0,1\}^n$ *and guesses* $\sigma_1 \ldots \sigma_k \in \{0,1\} [Hope: \exists!\ x \in L_f, \forall i \in [k], \sigma_i = x \odot s_i]$

So, in the correct code word we are assuming that the bit is σ_i which means if you recall the definition of Walsh Hadamard that the inner product of x and s_i is σ_i. So, let us make these k guesses what is it that you have in the correct code word of related to x WH x. So, make these k guesses and then based on this you do the queries if this guess is correct then you will be able to recover x that is the plan and 2^k notice 2^k is around n/ε^2 so that is supposed to be the list.

So, we are hoping that this identifies the unique x $\sigma_1 \ldots \sigma_k$ and define $s_T := \oplus_{i \in T} s_i$ & $\sigma_T := \oplus_{i \in T} \sigma_i$ (bit wise XOR) $\forall T \leq [k]$So, we have these locations now to be queried s $_T$th location f to be queried so let us query.

So,$compute\ x_i := maj_T\{\sigma_T \oplus f(s_t \oplus e_i)\}$

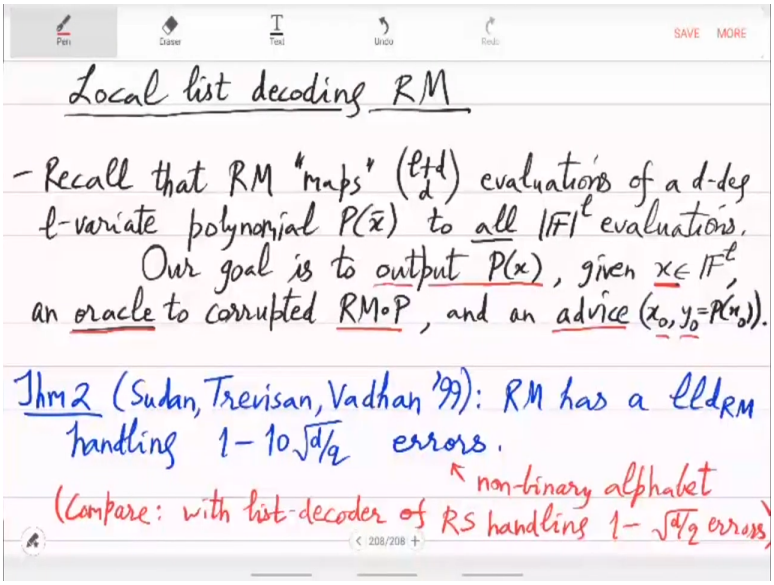
$OUTPUT: x_1 \ldots z_n$

So, if you repeat it enough then hopefully you will get the whole list that is the idea. So, repeat the above algo for let us say $1000n/\varepsilon^4 times$.so hopefully this gives you the full list L $_f$ so that is the local list decoder I would want to skip the analysis of this. But let me just say if you want to see the analysis in my old lecture notes on the homepage or you can see it in the textbook.

The main idea in the analysis would be to show that with the right guess when $\sigma_1 \ldots \sigma_k$, are correct when guesses are correct. Then this output is correct it is an L $_f$ with high probability that is what you have to show but there will be a lot of probability calculation in this. Obviously if the guess is not correct then you will get either so when the guess is incorrect then what happens then you may get the wrong answer but when the guesses correct then you get the x you actually identify a unique x in the list.

So, this will really depend on the guesses you can think of the guesses also as a part of the input. Hence it is a good local list a coder given the right guess, the right advice then actually you get an element in the list otherwise not that is what you have to show. So, now let us move to local list decoder for Reed Muller.

(Refer Slide Time: 21:08)



So, here again you have to recall what you did for **local decoding of Reed Muller**. So, you drew a random line passing through the point you are interested in and then use Reed Solomon decoding so we will do a similar thing now for local list decoding as well so recall that Reed Mueller maps $\frac{l+d}{d}$ evaluations of a d degree l variate polynomial $p(\bar{x})$ to all $|F|^l$evaluations. So, these given evaluations $\frac{l+d}{d}$ evaluations the message that is essentially as good as giving the coefficients.

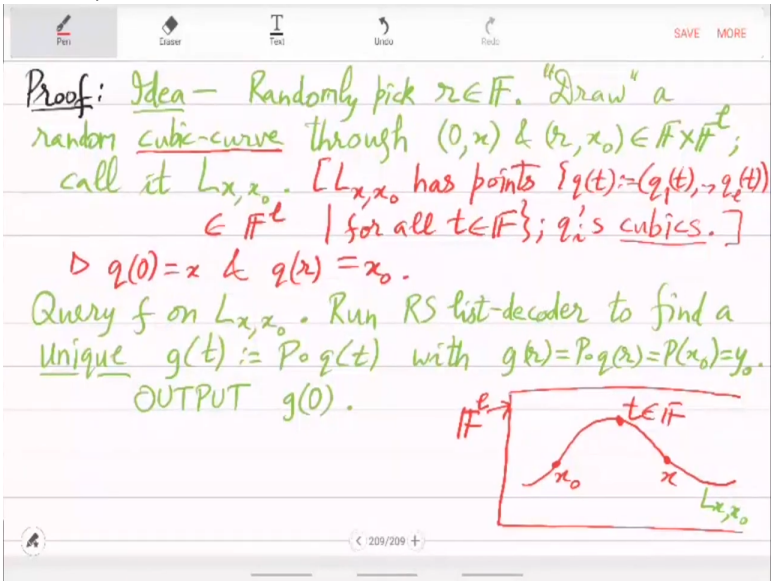
So, describing P uniquely and once p is described then the code word is all evaluations. So, there is redundancy and now when some of the evaluations when part of the code word gets corrupted how do you recover x or x_j. So, now our goal is to output P (x) given x $\epsilon |F|^l$ given a point in the space and oracle to a corrupted code word Reed Muller ◦ P and an advice. So, advice here will be a point in the space x_0 and its value that will be the advice $(x_0, y_0) = p(x_0)$so you are given a point in space.

And you are given an oracle to RM of P and you are given an advice string which is basically evaluation of P unknown P. Now there will be many P's what this advice will do is it will specify uniquely P that is why you need this advice and with all this you want to output P (x). So, given these 3 things in the input you want to evaluate P at x how do you do this? Remember that there are many P's possible because RM of P is corrupted so much close to 1/2.

In fact close to in this case it will be close to 100% close to 1. So, there will be many P's there is a list but this advice will pinpoint some P in the list and for that P you want to evaluate no matter what x is given P (x) this was done by **Sudan, Trevisan, Vadhan**. So, Reed Muller has a local list decoder handling close to 100% errors in the non binary alphabet. So, it is $1 - 10\sqrt{d/q}$ remember that this is the non binary version.

So, out of this F^l evaluations almost 100% field elements are erroneous the erroneous evaluations and the field is large the alphabet is also large that is given by the finite field. So, you can compare it with list decoder of Reed Solomon that handles$1 - 10\sqrt{d/q}$errors. So, in that sense there is some restriction here so Reed Solomon could handle more errors. Read Muller local list decoder is handling fewer but still it is comparable. It is in so bad it is in fact very good. So, again let us just look at the proof idea we cannot do the whole proof.

(Refer Slide Time: 28:33)



Idea :Randomly pick a point $r \epsilon F$so remember in local decoding of Reed Muller we drew a line. So that was degree 1 object now we will draw cubic curve instead of degree 1 curve now we will draw degree 3 curves. So, it is a cubic curve through(0,x) &$(r,x_{0.}) \epsilon F \times F^l$

Basically it is x and x_0 is given you want to associate x_0 and x_0 with something random which is r so random cubic curve through these 2 call it L_{x,x_0} it is a cubic curve. So, [L_{x,x_0} $has\ points\ \{q(t) := q_1(t), \ldots, q_l(t)\ ) | for\ all\ t \in F\};\ q_i's\ cubics$].

So, in this space you have drawn this parameterized cubic curve it parameterize by t and t takes all the values in the finite field and $q(0) = x$ & $q(r) = x_0$ remember this which we say that the cubic curve is passing through x and x 0. Now query f is the oracle given to you to the corrupted code word it is basically you know evaluations of p but many of them are corrupted close to half of them.

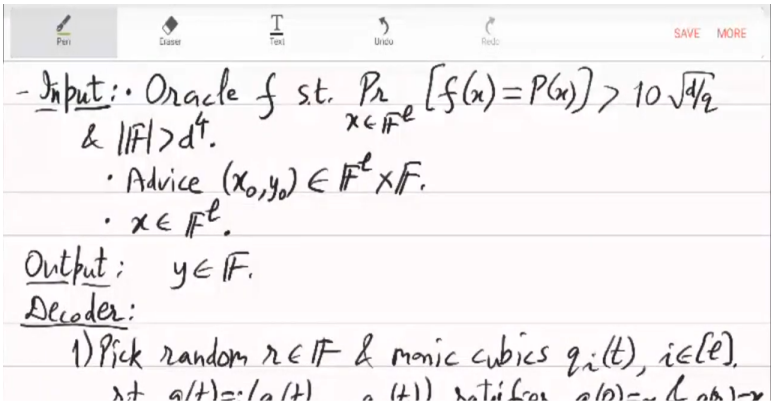
So, we will only look at those evaluations which are at the points of this curve L so query f on L_{x,x_0} run Reed Solomon list decoder to find a unique g(t). So, $g(t) = p \circ q(t)\ with\ g(r) = p \circ q(r) = p(x_0) = y_0$

$OUTPUT\ g(0)$

. That is the answer so you can output this. So, the picture that this requires is you have drawn a curve which has x_0 and it has x and it has this general t going over the fields field element.

And that is your L you are interested in basically it have to pass through 2 points and still have some freedom. So, this is why we needed degree 3 and remember that this happens in a finite space. So, this space is F^l all this happens in the space F^l this is a very discrete object but analytically you can think of this

(Refer Slide Time: 36:22)



Input : $Oracle\ f\ st\ Pr_{x \in F^l}[f(x) = p(x)] > 10\sqrt{d/q} \& |F| > d^4$.

Advice: $(x_0, y_0) \epsilon F^l \times F$

• $x \epsilon F^l$

OUTPUT : $y \epsilon F$

Decoder :

1) Pick random $r \epsilon F$ & $monic\ cubics\ q_i(t), i \epsilon [l]\ s.t\ q(t) = (q_1(t) \ldots\ldots q_l(t))\ satisfies\ q(0) = x \& q(r) = x_0$
2) Query f on L_{x,x_0} to obtain s:={$(t, f \circ q(t)) | t \epsilon F$}

. Now many of them may be wrong given the property of f many of them may be wrong.

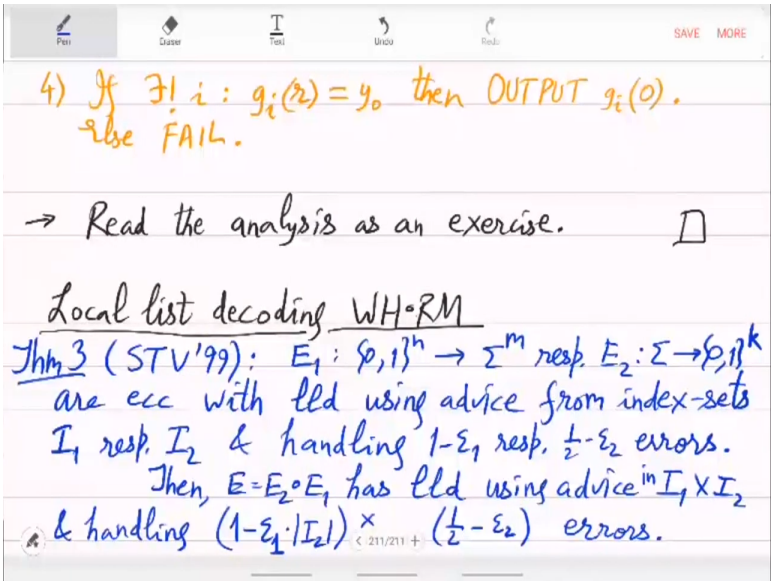
But then you can list decode using the Reed Solomon list decoder and with the input guarantee you will be able to get a list of all the piece out of which 1 will be identified because of this advice.

3) Run RS list decoder on s to find the list $g_1 \ldots . g_k$ of all deg-3d polynomials that agree $\geq 8\sqrt{dq}\ pairs\ in\ s$

so this is there is a good chance that this will happen given the guarantee on if there is a good chance also because the cubic curve was picked randomly there is a good chance that this

distribution will be there and you will actually correctly get $g_1 \ldots . g_k$ out of which one will be right.

(Refer Slide Time: 43:23)



So, you pick that one so

4) If $\exists!\ i: g_i(r) = y_0$ *then OUTPUT* $g_i(0)$. *else FAIL*

. So, you have to repeat or just output fail and exit but the probability of that will be less will be small.

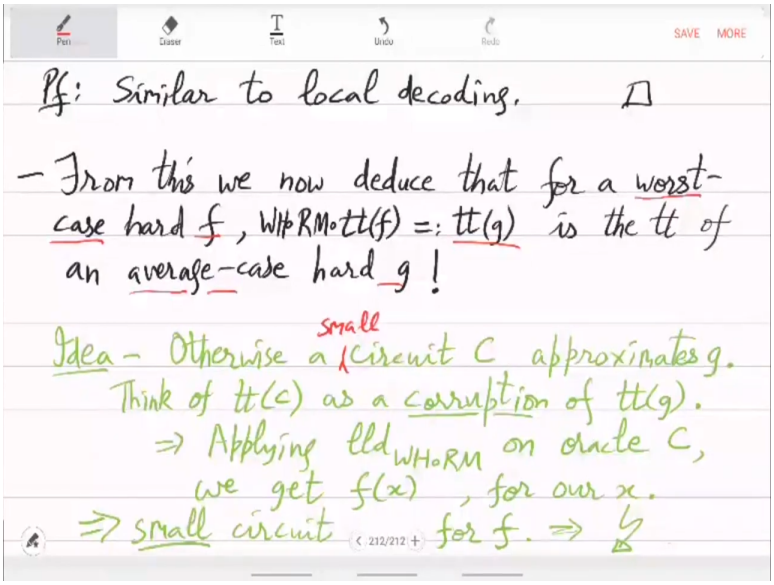
So, the analysis I am again skipping so read the analysis as an exercise.so that is essentially the **local list decoder for Reed Muller** drawing a cubic random cubic curve and then doing Reed Solomon list decoding. Finally we combine the 2 so again it is a **theorem by Sudan, Trevisan, Vadhan**: $E_1: \{0,1\}^n \to \Sigma^m$ *rsp* $E_2: \{0,1\}^k$ *ar ecc with lld using advice from index* $-$ *sets* I_1 *resp* I_2 & *handling* $1 - \varepsilon_1$ *resp* $1/2 - \varepsilon_2$ *errors.*

Then $E = E_2 \circ E_1$ *has lld using advice* $I_1 \times I_2$ *and handing* $(1 - \varepsilon_1|I_2|) \times (1/2 - \varepsilon_2)$*errors.*

. So, when you apply the local list decoder for E 2 you have to pay more so you will pay this factor of I 2 so it will be this so actually to get what advice to pick from I 2 you will try all the advice actually.

So, as many advice strings there are in I 2 so that price you pay and for the binary it is $1/2 - \varepsilon_2$ and then you take the product this proof is actually similar to the local decoder and I will completely skip that.

(Refer Slide Time: 49:37)



Similar to local decoding so finally we are finished local list decoding for all these maps what do we have now? So, now all that remains is to apply this to get average case hard function. So, from this we now deduce that for a worst case hard f , $WH \circ RM \circ tt(f) =: tt(g)$

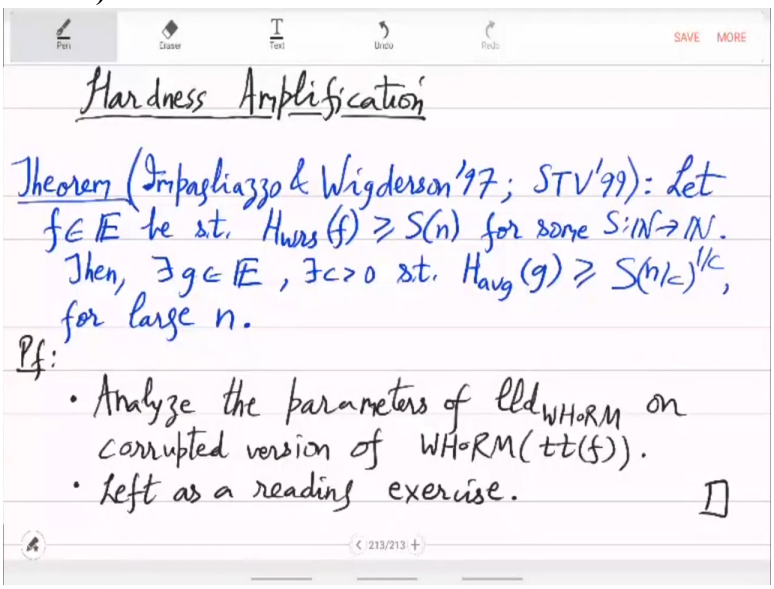
So, this is the truth table of an average case hard function g !why that intuitively because if g is not average case hard then there is a circuit which can compute this on significantly more than half of the input cases. So, now the truth table of that circuit you can think of as a corrupted code word and from that corrupted code word when you apply the local list decoder that we have just built then you will can evaluate f (x) that is the idea.

Otherwise the circuit c approximates g and think of the tt(c) of is a corruption of tt g truth table of c you have a oracle access. So, now apply $lld_{WH\circ RM}$ on this oracle c we get f(x)now the c is an oracle it is a circuit small circuit in fact and the local list decoder is an algorithm which is very fast.

So that also is a can think of yourself small circuit. So, overall you get circuit for f which is a contradiction that is the idea. So, in the proof details you have to look at the algorithm local list

decoder and convert it into a circuit and check the parameters that you really get a polynomial sized circuit moreover it is the advantage over half should be significant. So, all those things you have to are well that want showed of should not be significant in fact because you want to prove that g is average case hard. So, those things you have to look into the details so let me just state the theorem and then stop.

(Refer Slide Time: 56:10)



Hardness amplification so this theorem is by **Impasliazzo & Wigderson** they proved it first in 97 by combinatorial methods and then this error correcting code based proof was given by STV in this paper that we have been talking about Sudan, Trevisan, Vadhan from 99 this was an algebraic proof to amplify the hardness. So,

Impasliazzo & Wigderson: $Let\ f \epsilon E, \exists c > 0\ s.t\ H_{avg}(g) \geq S(n/c)^{1/c}\ for\ large\ n$ So, it is much harder than f it is in the same complexity class E we will do not have time to do the proof of this but it is based on just applying the local list decoder.

So, analyze the parameters of local list decoder of Walsh Hadamard, Reed Mullar, on corrupted version of Walsh Hadamard Reed Muller applied on the truth table f that is the code word and then it is corrupted and the local list decoder wants to get back to f some location in the string tt f. So, you have to analyze the whole thing all the parameters and you will get this so, left as an exercise so that finishes a major theorem in complexity theory. And with that the course ends I hope you enjoyed the course. Thank you.