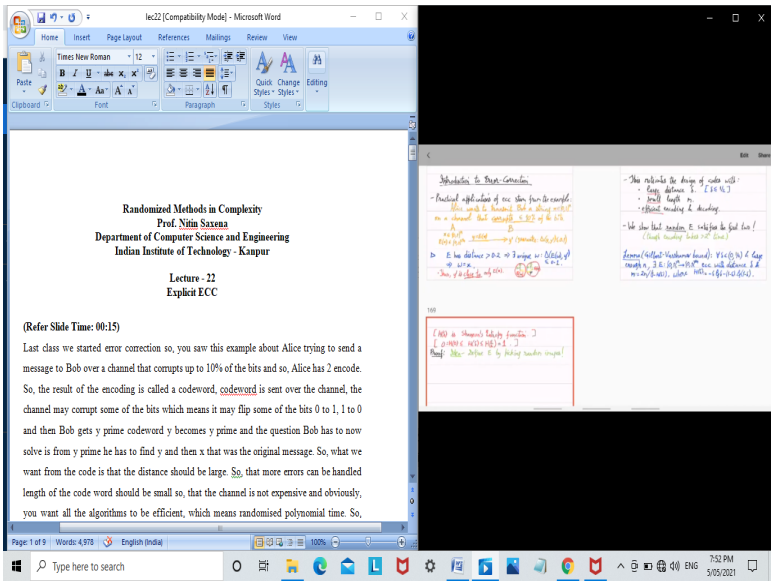
Randomized Methods in Complexity
Prof. Nitin Saxena
Department of Computer Science and Engineering
Indian Institute of Technology - Kanpur

Lecture - 22
Explicit ECC

(Refer Slide Time: 00:15)

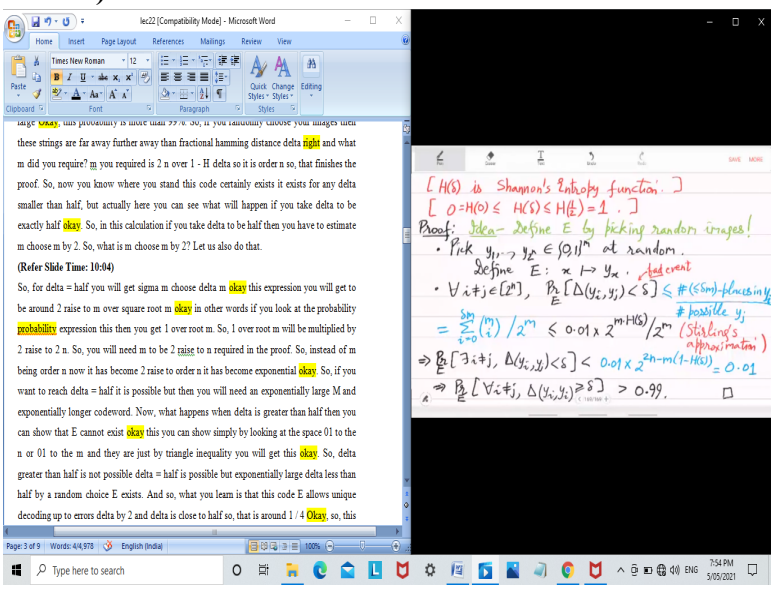


Last class we started error correction so, you saw this example about Alice trying to send a message to Bob over a channel that corrupts up to 10% of the bits and so, Alice has 2 encode. So, the result of the encoding is called a codeword. Codeword is sent over the channel, the channel may corrupt some of the bits which means it may flip some of the bits 0 to 1, 1 to 0 and then Bob gets y' codeword y becomes y' and the question Bob has to now solve is from y'he has to find y and then x that was the original message.

So, what we want from the code is that the distance should be large. So, that more errors can be handled length of the code word should be small so, that the channel is not expensive and obviously, you want all the algorithms to be efficient, which means randomised polynomial time. So, first we will look at the existential state what is the best distance and the length that you can hope for that is given by Gilbert Varshamov bound.

So, the theorem says that or this lemma states that there is a code which will basically give every string in 0, 1 to the n a random m bit code word where the distance will be delta which you can pick to be any constant between 0 and half.

(Refer Slide Time: 01:56)



So, let us show this so, we will basically just have to randomly pick the images the code words. So, pick $y_1,\ldots,y_{2^n}\epsilon\{0,1\}^m$ at random and define $E\colon x \mid \rightarrow y_x$ So,2^n strings are mapped to 2^n strings, but these images are longer that is n bits. Now, let us look at the distance between y_x and y_x' or y_i and y_j for every i j pair different we are interested in this

$$\forall i \neq j \epsilon [2^n], Pr_E[\Delta(y_i, y_j) < \delta]$$

So, over the choices that he makes, what is the probability that fractional hamming distance between y_i, y_j is less than δ this will be the bad event. So, this bad event what is the probability? This probability is less than equal to you can think of y_i is already fixed. So, I mean y_i this image these you are picking both y_i and y_j randomly, but you can assume y_i to be fixed and then y_j is what you are picking.

So, what are the good y_j's over all possible y_j's. So, number of y_j's which are close to y_i, so, $\forall i \neq j \epsilon [2^n], Pr_E[\Delta(y_i, y_j) < \delta] \leq \frac{\#(\leq \delta_m) - places\ in\ y_i}{\#possible\ y_j}$. So, if you want y_j close to y_i then you

basically have to pick δ_m locations in y_i and just flip them. So, every this subset choice will give you a bad y_j and divided by the total number of y_j's possible.

So, $\forall i \neq j \epsilon [2^n], Pr_E[\Delta(y_i, y_j) < \delta] \leq \frac{\#(\leq \delta_m) - places\ in\ y_i}{\#possible\ y_j} = \sum_{i=0}^{\delta_m} \frac{m}{i} / 2^m$. Now, basically this binomial sum is what you have to estimate. So, you can see that $\frac{m}{\delta m}$, which is the largest summoned here that will be much smaller than 2^m. So, that is given by sterling's approximation, so, you can actually write it like $\sum_{i=0}^{\delta_m} \frac{m}{i} / 2^m \leq 0.01 \times 2^{m.H(\delta)} / 2^m$

So, this is really what you get you get that the binomial sum is much smaller than 2^m and this exponent is $H(\delta)$ given by the Shannon's entropy function this is really coming from Sterling's approximation. So, you can look this proof of Sterling's approximation it basically gives you a very good asymptotic bound for m factorial and then using that you can get bounds for binomial numbers and then you can also get bounds on some of binomials.

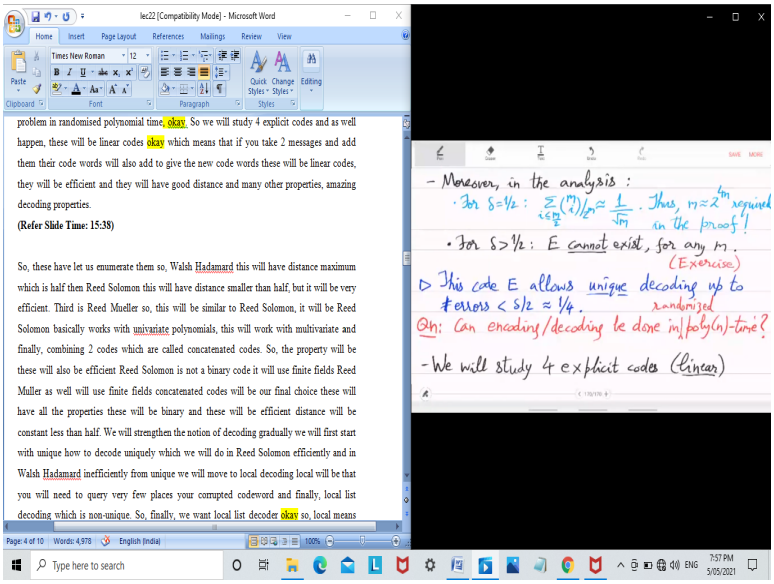
So, if you sum up, up to δm these binomial numbers then you get a value smaller than $2^{m.H(\delta)}$. So, if you take delta to be half then this is square root of 2^m /2 that is the point. So, that is the probability that if you randomly pick y_i, y_j, they will be close the distance will be smaller than delta, but we are interested in all the pairs i j what is the probability for that? So, that you can now; reduce so, $Pr_E[\exists i \neq j, \Delta(y_i, y_j) < \delta] \leq 0,01 \times 2^{2n - m(1 - H(\delta))} = 0.01$, but still if you take m to be quite large, appropriately large then this fraction is very small this probability is very small. In fact, at this point you should see what m is we have fixed in the lemma m to be to $2n / 1 - H(\delta)$.

So, you already get that this is 0.01 and that means that $Pr_E[\exists i \neq j, \Delta(y_i, y_j) \geq \delta] > 0.99$. So, if you randomly choose your images then these strings are far away further away than fractional

hamming distanceδ and what m did you require? m you required is 2n / $1 - H(\delta)$ so it is order n so, that finishes the proof.

So, now you know where you stand this code certainly exists it exists for anyδ smaller than half, but actually here you can see what will happen if you take δ to be exactly half. So, in this calculation if you takeδ to be half then you have to estimate$\frac{m}{m/2}$. So, what is $\frac{m}{m/2}$? Let us also do that.

(Refer Slide Time: 10:04)



$For\ \delta > 1/2,\ \sum_{i \le m/2} \frac{m}{i}/2^m \approx \frac{1}{\sqrt{m}}$Thus $m \approx 2^{4n}$required in th proof

So, if you want to reach $\delta = 1/2$it is possible but then you will need an exponentially large m and exponentially longer codeword. Now, what happens when$\delta > 1/2$ then you can show that E cannot exist this you can show simply by looking at the space$\{0,1\}^n$ or $\{0,1\}^m$and they are just by triangle inequality you will get this. So, $\delta > 1/2$ is not possible $\delta = 1/2$is possible but exponentially large delta less than half by a random choice E exists.

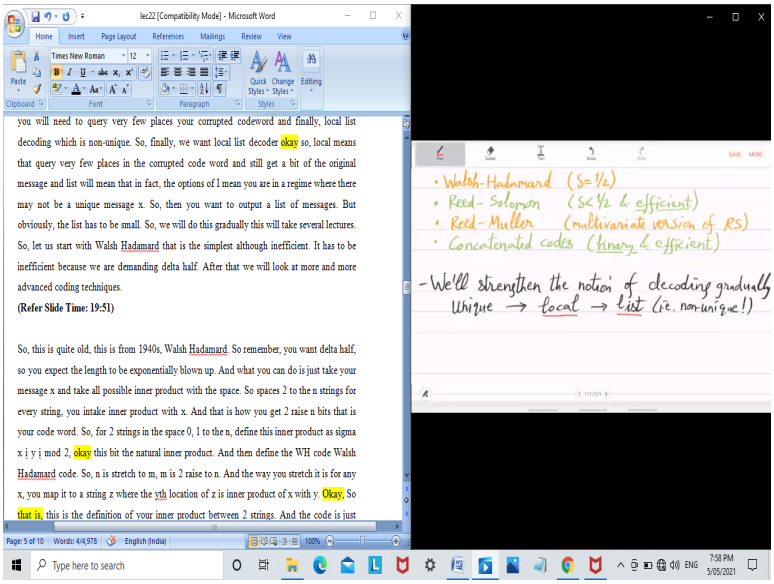
And so, what you learn is that this code E allows unique decoding up to errors< $\delta/2 \approx 1/4$ so, this existential code E he tells you that all information theoretically unique decoding is possible

even if there is 25% error. So, Alice send this message or this code word to Bob and 25% of the string was corrupted still Bob can recover the original message so, this is an amazing thing.

The only part missing is can this be done efficiently? Can encoding decoding be done in poly(n) time? So, encoding can be done in this existential proof it can be done 2^n time because it basically goes through all the string$\{ 0,1\}^n$. So, can you make it poly(n)- time and also randomised is allowed. So, can you solve this problem in randomised polynomial time.

So we will study 4 explicit codes and as well happen, these will be linear codes which means that if you take 2 messages and add them their code words will also add to give the new code words these will be linear codes, they will be efficient and they will have good distance and many other properties, amazing decoding properties.

(Refer Slide Time: 15:38)



So, these have let us enumerate them so, **Walsh Hadamard** ($\delta = 1/2$)this will have distance maximum which is half then **Reed Solomon** ($\delta < 1/2$ & $efficient$)t. Third is Reed Mueller so, this will be similar to Reed Solomon, it will be **Reed Solomon** (multivariate version of RS)

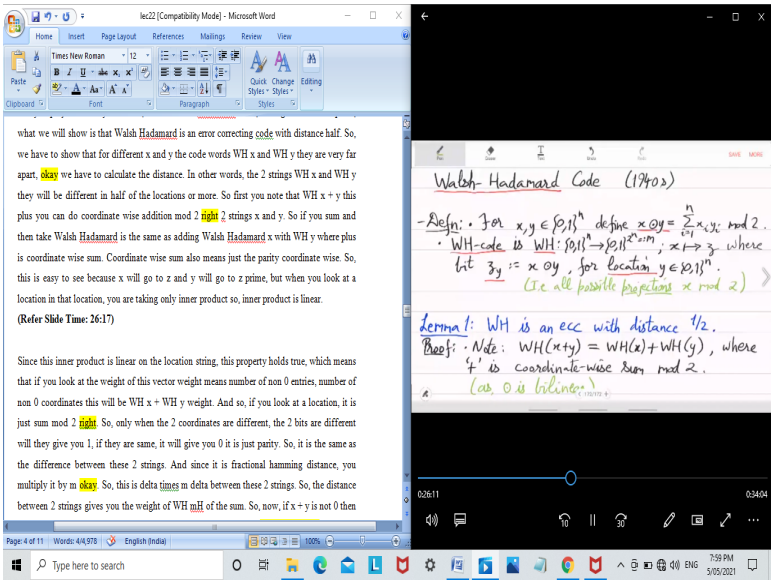
So, the property will be these will also be efficient Reed Solomon is not a binary code it will use finite fields Reed Muller as well will use finite fields **concatenated codes**(binary & efficient). We will strengthen the notion of decoding gradually we will first start with unique how to decode

uniquely which we will do in Reed Solomon efficiently and in Walsh Hadamard inefficiently from unique.

We will move to local decoding local will be that you will need to query very few places your corrupted codeword and finally, local list decoding which is non-unique. $unique \rightarrow local \rightarrow list(i.e\ non\ -\ unique!)$So, finally, we want local list decoder so, local means that query very few places in the corrupted code word and still get a bit of the original message and list will mean that in fact, the options of I mean you are in a regime where there may not be a unique message x. So, then you want to output a list of messages. But obviously, the list has to be small.

So, we will do this gradually this will take several lectures. So, let us start with Walsh Hadamard that is the simplest although inefficient. It has to be inefficient because we are demanding delta half. After that we will look at more and more advanced coding techniques.

(Refer Slide Time: 19:51)



Walsh Hadamard code(1940). So remember, you want delta half, so you expect the length to be exponentially blown up. And what you can do is just take your message x and take all possible inner product with the space. So spaces 2^n strings for every string, you intake inner product with

x. And that is how you get 2^n bits that is your code word. So, for 2 strings in the space $\{0,1\}^n$,

defn: $For\ x, y \epsilon \{0,1\}^n\ define\ x \odot y = \sum_{i=1}^{n} x_i y_i mod 2$

And then define the **WH code (Walsh Hadamard code)**.

$WH: \{0,1\}^n \to \{0,1\}^{2^n =: m}\ ;\ x \to z\ where\ z_y := x \odot y\ , for\ location\ y \epsilon \{0,1\}^n$

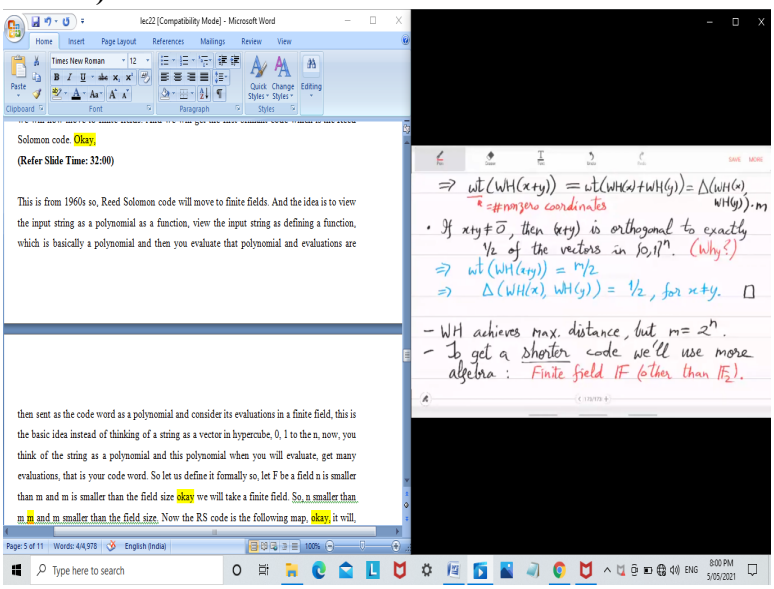
So it is a 2^n length string, you can also think of this as projections of x all possible projections modulo 2 you project inner product is also like projections but this is mod 2 so there are 2^n projections possible and you project in every direction, that is Walsh Hadamard code, what good is this map. So, what we will show is

Lemma 1: Walsh Hadamard is an error correcting code with distance 1/2.

So, we have to show that for different x and y the code words WH x and WH y they are very far apart, we have to calculate the distance. In other words, the 2 strings WH (x) and WH (y) they will be different in half of the locations or more. $WH(x + y) = WH(x) + WH(y)$ where '+' is coordinate wise sum mod2. Coordinate wise sum also means just the parity coordinate wise.

So, this is easy to see because x will go to z and y will go to z', but when you look at a location in that location, you are taking only inner product so, inner product is linear.

(Refer Slide Time: 26:17)



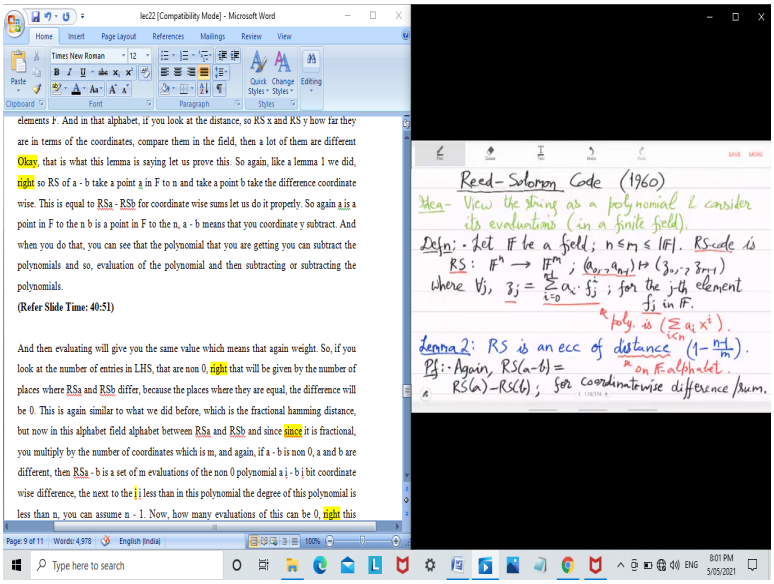
Since this inner product is linear on the location string, this property holds true, which means that if you look at the weight of this vector weight means number of non 0 entries, number of non 0 coordinates this will be wt(WH(x+y))=wt(WH(x)+WH(y)). And so, if you look at a location, it is just sum mod 2. So, only when the 2 coordinates are different, the 2 bits are different will they give you 1, if they are same, it will give you 0 it is just parity. So,wt(WH(x+y))=wt(WH(x)+WH(y))= Δ(WH(x),WH(y)).m. So, If $x+y \neq \bar{0}$,then(x+y) is orthogonal to exactly ½ of the vectors in $\{0,1\}^n$

So, a non 0 vector is orthogonal to exactly half of the space and it is not orthogonal to half of the space. So, using that property you know that weight of this Walsh Hadamard x + y is half the weight of the Walsh Hadamard because Walsh Hadamard vector is nothing but collection of these inner products, half of them is 0 half of them is non 0. So, you get wt(WH(x+y))=m/2, Δ(WH(x),WH(y))=½ for $x \neq y$

Lemma 1, you have to show that Walsh Hadamard is an error correcting code with distance half. And we have shown that indeed, if you look at the images, the code words they are far away. And moreover, it is a linear code word, but it is very long so, we want shorter code words. So, WH achieves maximum distance, but this m is exponential $m=2^n$ So now, to get shorter code we have to do something else to get a shorter code we will look at finite fields so we will use more algebra.

So finite fields F other than F_2 you can see that Walsh Hadamard work with the finite field F_2 field with 2 elements it did addition mod 2. But inspired by this, we will now move to finite fields. And we will get the first brilliant code which is the Reed Solomon code.

(Refer Slide Time: 32:00)



This is from 1960s so, **Reed Solomon code** will move to finite fields. And the idea is to view the input string as a polynomial as a function, view the input string as defining a function, which is basically a polynomial and then you evaluate that polynomial and evaluations are then sent as the code word as a polynomial and consider its evaluations in a finite field, this is the basic idea instead of thinking of a string as a vector in hypercube, $\{0,1\}^n$.

Now, you think of the string as a polynomial and this polynomial when you will evaluate, get many evaluations that is your code word. So let us define it formally so, let F be a field $n \leq m \leq |F|$Now the RS code is the following map, it will, instead of a string it will actually take a point in$F^n \to F^m$; $(a_0,\ldots,a_{n-1}) \to (z_0,\ldots,z_{m-1})$ where $\forall j, z_j = \sum_{i=0}^{n-1} a_i f_j^i$; $for\ the\ j-\ th\ element\ f_j\ in$ F so, that is the RS code.

RS code is mapping a point is a $(a_0,\ldots,a_{n-1})$ by defining a polynomial and then evaluating it at f_j so, we can write that down. So, the polynomial is $\sum_{i<n}(a_i X^i)$ this is the polynomial that the given point$(a_0,\ldots,a_{n-1}$ defines and you just evaluate this polynomial at many, many field elements, exactly m field elements. Now again notice that this is a linear code, because if you take 2 points.

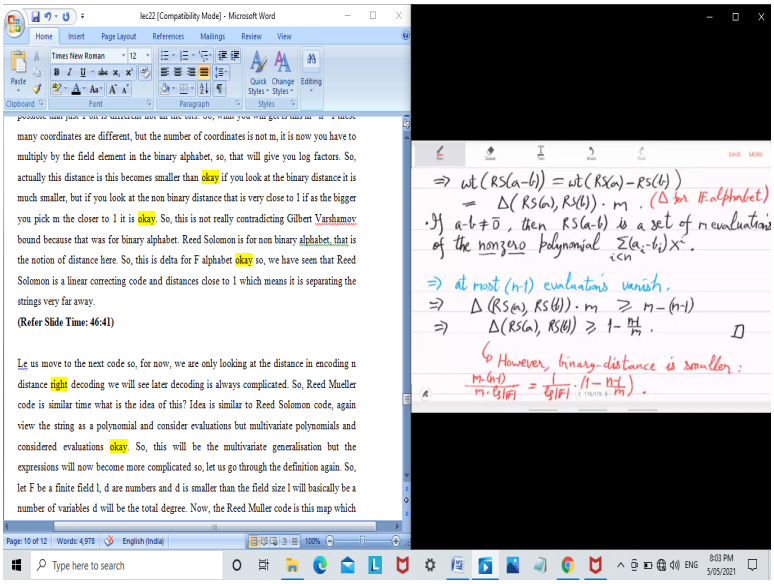
And look at the RS code words, when you sum the 2 points again coordinate wise and in the finite field then the code words will also add up this is because the 2 polynomials you can add, you can add the 2 polynomials and then evaluate or you can separately evaluate and add so this is again a linear code. And the property was what is the distance of this? So RS is an error correcting code of distance so if you take m to be more than double that of n, we are claiming that the distance of RS is half.

Lemma 2:RS is an ecc of distance $(1 - \frac{n-1}{m})$

Actually, you should be careful here, the distance notion here is for the finite field alphabet this is not binary anymore. So let us remember that so the alphabet is now not 0, 1 but the field elements F. And in that alphabet, if you look at the distance, so RS x and RS y how far they are in terms of the coordinates, compare them in the field, then a lot of them are different that is what this lemma is saying let us prove this.

So again, like a lemma 1 we did, so RS(a-b)=RS(a) -RS(b) for coordinate wise sums let us do it properly. So again a is a point in F^n ,b is a point in F^n, (a - b) means that you coordinate y subtract. And when you do that, you can see that the polynomial that you are getting you can subtract the polynomials and so, evaluation of the polynomial and then subtracting or subtracting the polynomials.

(Refer Slide Time: 40:51)



And then evaluating will give you the same value which means that again weight. So, if you look at the number of entries in LHS that are non 0, that will be given by the number of places where RS(a) and RS(b) differ, because the places where they are equal, the difference will be 0. This is again similar to what we did before, wt (RS(a-b))=wt (RS(a)-RS(b))= $\Delta(RS(a), RS(b)).m$ and again, if a - b $\neq \overline{0}$, a and b are different, then RS(a - b)is a set of m evaluations of the $\sum_{i<n}(a_i - b_i)X^i$, the degree of this polynomial is less than n, you can assume n - 1. Now, how many evaluations of this can be 0, this maximum n - 1 evaluation are 0 so, remaining evaluations are non 0.

So, at most (n - 1) evaluations vanish which means that $\Delta(RS(a),RS(b)).m \geq m-(n-1)$

$$\Delta(RS(a),RS(b)) \geq 1-(n-1)/m$$

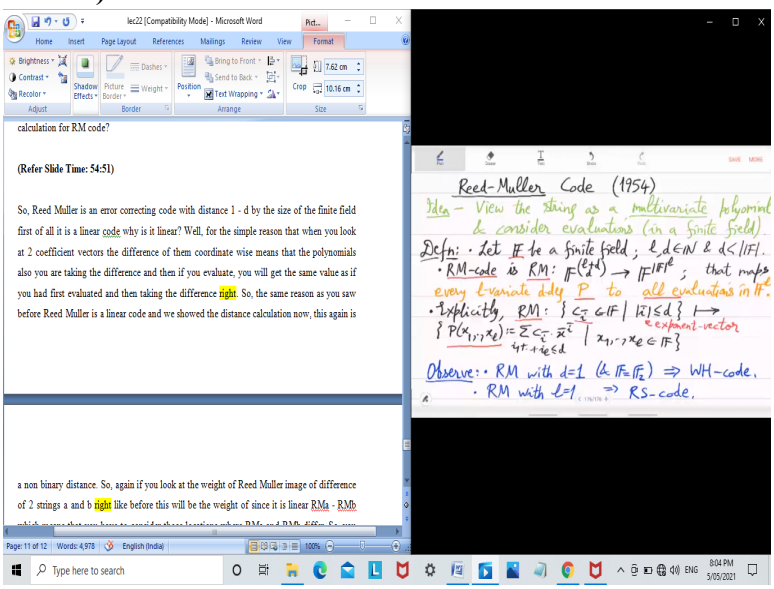
which is what we wanted to show however, if you look at the binary distance so, for binary distance you have to actually look at the coordinates in binary.

Now, when you look at field alphabet 2 field elements are different, but that only means that when you look at the binary representation 1 bit is different, it is possible that just 1 bit is different not all the bits. So, what you will get is this m -(n - 1) these many coordinates are

different, but the number of coordinates is not m, it is now you have to multiply by the field element in the binary alphabet, so, that will give you log factors.

binary distances are smaller: $\frac{m-(n-1)}{m.log|F|} = \frac{1}{log|F|}.(1-\frac{n-1}{m})$ if you look at the binary distance it is much smaller, but if you look at the non binary distance that is very close to 1 if as the bigger you pick m the closer to 1 it is. So, this is not really contradicting Gilbert Varshamov bound because that was for binary alphabet. Reed Solomon is for non binary alphabet that is the notion of distance here. So, this is delta for F alphabet so, we have seen that Reed Solomon is a linear correcting code and distances close to 1 which means it is separating the strings very far away.

(Refer Slide Time: 46:41)



Let us move to the next code so, for now, we are only looking at the distance in encoding n distance decoding we will see later decoding is always complicated. So, **Reed Mueller code** is similar time what is the idea of this? Idea is similar to Reed Solomon code again view the string as a polynomial and consider evaluations but multivariate polynomials and considered evaluations.

So, this will be the multivariate generalisation but the expressions will now become more complicated so, let us go through the definition again. **Defn**;So, let F be a finite field

$l, d \in N$ & $d < |F|$ Reed Muller code is RM:$F^{\frac{l+d}{d}} \to F^{|F|^l}$ that maps every l variate d degree polynomial P to all evaluations in the space F^l if for every point in the space F^l you evaluate this P and hence you will get F^l many coordinates that is your code word.

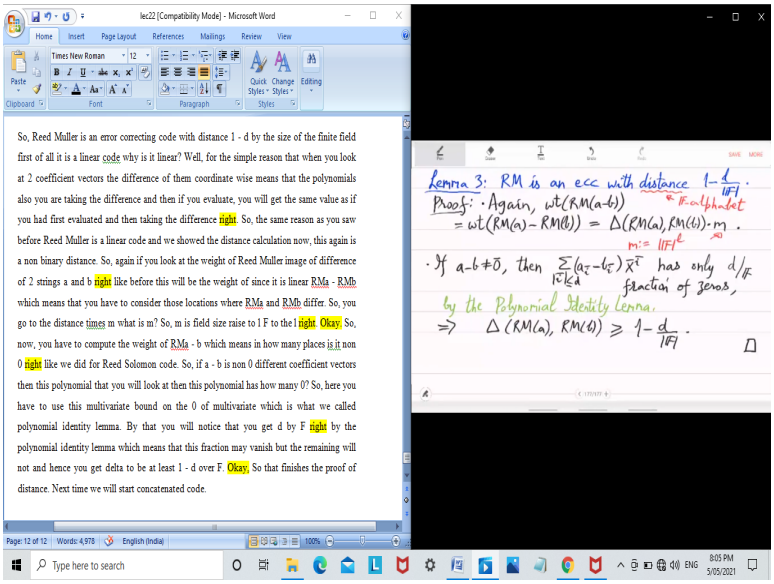
So, explicitly also we can write this RM : $\{c_{\bar{i}} \in F | \, |\bar{i}| \leq d\} \to \{p(x_1, \ldots, x_l) := \sum_{i_1 + \ldots i_l \leq d} c_{\bar{i}} \bar{x}^{\bar{i}} \, | x_1 \ldots . x_l \in F\}$

So, you start with coefficients and you end up with evaluations of this polynomial P. So, I hope this clears up what the RM map is doing so, first observation is let us compare this with the Reed Solomon definition and Walsh Hadamard. So, both the codes that we saw before in Walsh Hadamard you can think of d = 1. So, RM with d = 1 if you make degree to be 1, then you are basically looking at a linear polynomial.

And if you also fix the finite field to be F_2 then what you are doing is given these $l$bit string you are just evaluating all these linear I mean you are evaluating this linear polynomial which$l$ bit string defines and hence you are just computing the inner product in all possible ways, all projections so you actually get the WH code. Now, to get Reed Solomon, you just fix$l$ = 1, that is natural.

If you take a univariate version of RM code, then this map is just the definition of Reed Solomon. So, this is why Reed Muller code is important it is actually a uniform generalisation of both these important codes that you saw Walsh Hadamard with delta, which is a binary code with distance 1/2 and Reed Solomon which is a non binary code with distance almost 1 and what is the distance calculation for RM code?

(Refer Slide Time: 54:51)



So, Reed Muller is an error correcting code with distance $1 - \frac{d}{|F|}$ first of all it is a linear code why is it linear? Well, for the simple reason that when you look at 2 coefficient vectors the difference of them coordinate wise means that the polynomials also you are taking the difference and then if you evaluate, you will get the same value as if you had first evaluated and then taking the difference.

So, the same reason as you saw before Reed Muller is a linear code and we showed the distance calculation now, this again is a non binary distance. So, again if you look at the weight of Reed Muller image of difference of 2 strings $wt(RM(a - b)) = wt(RM(a) - RM(b)) = \Delta(RM(a), RM(b)).m$.

what is m? So, m=$|F|^l$. So, now, you have to compute the weight of RM(a - b) which means in how many places is it non 0 like we did for Reed Solomon code. So, if a - b is non 0 different coefficient vectors then this polynomial that you will look at then this polynomial $\sum_{|\bar{i}| \leq d} (a_{\bar{i}} - b_{\bar{i}})\overline{X}^{\bar{i}}$ has only d/F fractions of zeros,by the polynomial identity lemma which means that this fraction may vanish but the remaining will not and hence you get

$\Delta(RM(a), RM(b)) \geq 1 - \frac{d}{|F|}$ So that finishes the proof of distance. Next time we will start concatenated code.