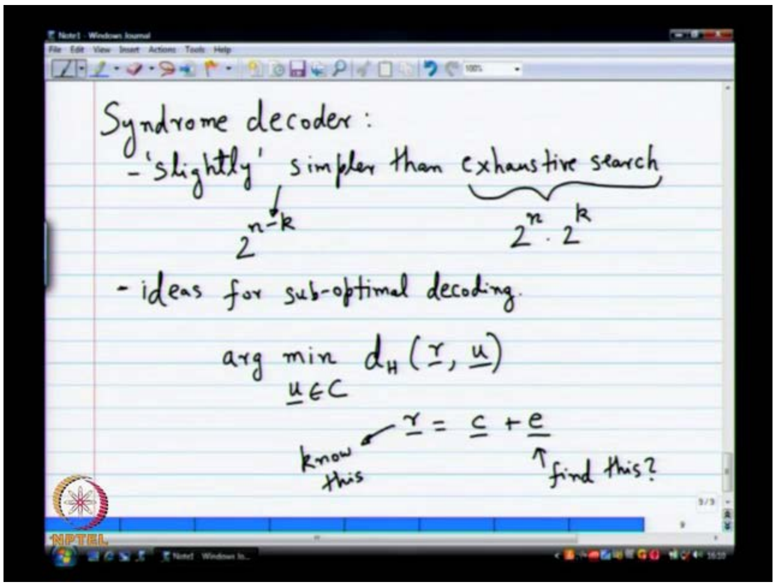
Coding Theory
Prof. Dr. Andrew Thangaraj
Department of Electronics and Communication Engineering
Indian Institute of Technology Madras

Lecture - 8
Syndrome Decoder, Basic of Finite Fields

(Refer Slide Time: 00:13)



Syndrome decoder:

- 'slightly' simpler than $\underbrace{\text{exhaustive search}}_{2^n \cdot 2^k}$

$\downarrow$ 2^{n-k}

- ideas for sub-optimal decoding.

$$\arg\min_{\underline{u} \in C} d_H(\underline{r}, \underline{u})$$

$$\underline{r} = \underline{c} + \underline{e}$$

know this

find this?

So, like I said, the exhaustive search to find the optimal decoder table is, looks to be exponentially complex in n and k. And there is a slightly simpler method like I said, which uses the syndrome decoder, slightly interesting idea, the complexity is 2 power and n minus k and for small examples it is interesting. And even for larger examples, it provides ideas for sub optimal decoding. So, that is the main point of syndrome decoder and it crucially depends on the parity check matrix and it turns out using the parity check matrix for decoding is far far better idea and in practice.

So, almost all the good codes are said decoded using the parity check matrix only, alright. So, what is the idea here? So, so what did we, what did we see was the optimal decoders. So, we wanted to do this operation, we wanted to do argument minimization u and c the having distance between r and u. So, this is the, this is the idea behind the optimal decoder. So, notice r is c plus e, you can write it as c plus e. So, we know r so, given r we have been trying to find the best possible c. So, notice was the prior probabilities for c, c is equally likely to be any one of the 2 power k possible code words.

You do not have any bias in the possible values for c ahead of time, we have no idea and we have to guess all the possibilities, if you are guessing c directly.

On the other hand, what is the totally equivalent is instead of finding c, you can as fell find e. So, if you try to find the error vector that was introduced by the channel, it is equivalent to trying to find the code word that was send, why is that?

Student: ((Refer Time: 02:36))

Yeah is r is equal to c plus e, I wrote that equation down. So, you can ever find c or you can find e, in fact when you find c here also finding an estimate of the error that happened in the chart. So, the question is, can you try and find this, how about finding this?

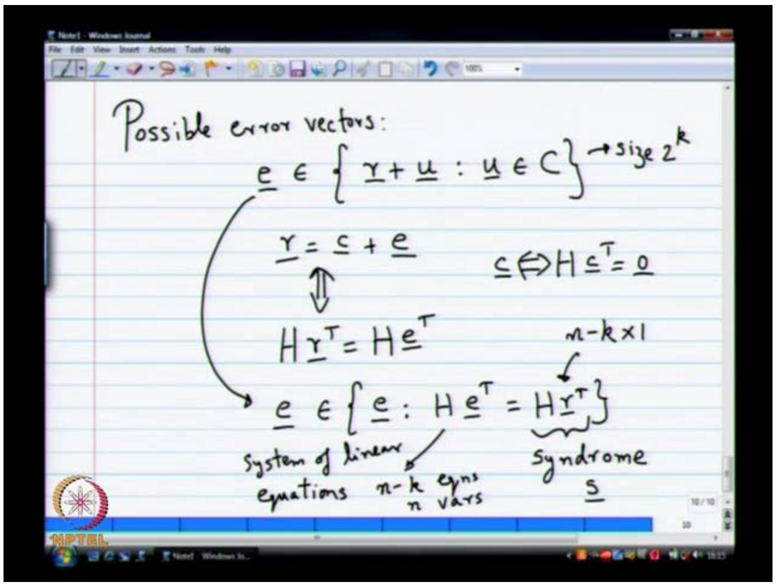
Student: ((Refer Time: 02:55)) to 0.

Yeah so, what is the advantage of finding this, trying to find this? The prior probabilities are not all equally uniform, will lightly to be, all the 2 power n possible vectors. So, at outside it seems like it is a bad idea, because c when there is only 2 power k possibilities, for e there are 2 power n possibilities, but it is heavily biased for every, any reasonable p it is very heavily biased. So, all zero vectors] is most likely and everything else is going to fall in likely hood. So, after a particular weight the probability of those things happening is very very less. So, you do not have to look at those things.

So, that is the thing that we will exploit in the syndrome decoder, try to find the error that happened that as opposed to the code word that was transmitted. And exploit the probabilities in the channel and that gives you a benefit. So, that is a basic idea, but how do you find the e, it seems like a complicated problem, right. The code is given to you, you know the code so, you just go to the code book and look for c, that I seem to know, but how do I look for the possible e.

So, as it turns out given an r you do not have all the 2 power n possibilities for e, yes or no? There will be only 2 power k because only 2 power k was transmitted, given that you received a particular r, there should be only 2 power k possibilities for e. I mean those only 2 power k starting points, given that you received anything, you cannot have all possible error vectors, you can only go back to one of the code words, only 2 power k possibilities exist. What is one way of describing the 2 power k possibilities?

(Refer Slide Time: 04:31)



Possible error vectors:

$$\underline{e} \in \{\underline{r} + \underline{u} : \underline{u} \in C\} \rightarrow \text{size } 2^k$$

$$\underline{r} = \underline{c} + \underline{e} \qquad \underline{c} \Leftrightarrow H\underline{c}^T = \underline{0}$$

$$\Updownarrow$$

$$H\underline{r}^T = H\underline{e}^T \qquad n-k \times 1$$

$$\underline{e} \in \{\underline{e} : H\underline{e}^T = \underbrace{H\underline{r}^T}_{\substack{\text{Syndrome} \\ \underline{s}}}\}$$

System of linear equations

n-k eqns
n vars

This is the, this is the way to describe it. What is the possible e, possible error vectors? So, you can say e belongs to r plus u such that, u belongs to the code c. This is a way of describing all possible error vectors, this is a way of describing the set of all possible error vectors. So, I know r is c plus e so, e should be of form r plus c for some code word c. So, simply write it as r plus u, u belonging to the code c, this is the way of describing the possible error vectors, this is one way of doing it. It returns out there is an equivalent way which is, which is really very powerful.

So, remember r is c plus e, right c satisfies an equation using the parity check matrix. So, h is the parity check matrix, you know c satisfies h times c transpose 0, not only that you also know that if there is arbitrary vector x. If H times x transpose is 0 then x is in the code, if it is not 0 then it is not in the code that also you know. So, this is kind of like an if and only if, goes both ways. So, if H times c transpose is not 0 then definitely c is not a code word, that vector is not a code word. So, when you use the relationship here, you can multiply throughout by H, you can take h and multiply by transpose of the left hand side and the transpose of the right hand side. And this equation becomes equivalent to H times r transpose equals H times e transpose.

Now using this property, one can see that the set of all possible e is actually let say, the set of all e such that H times e transpose equals H times r transpose. So, there are several ways of gregariously proving it, I do not want to write a big proof for the statements, it is quite easy to intuitively argue and you can write down a proof for this, it is not too difficult.

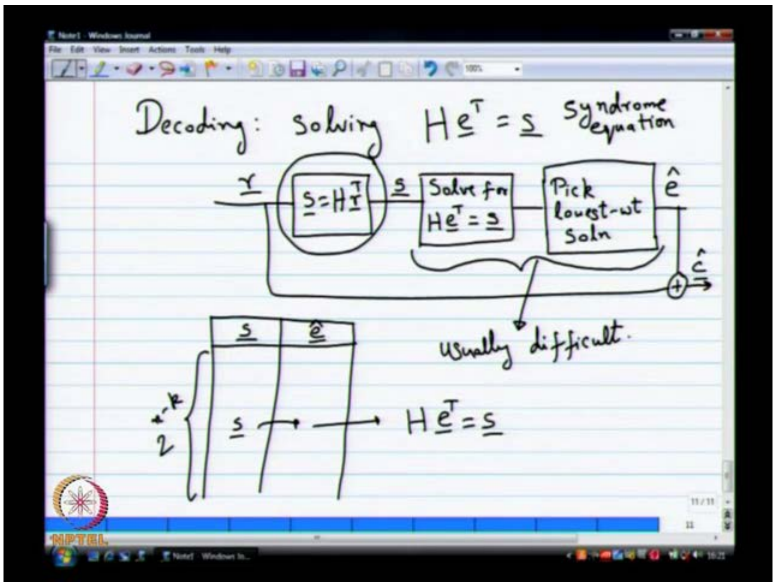
Student: But you cannot do so much, either the all zero or the particular...

Yeah, it is the set, sub set e belongs to the set, I will come back to this. So, what is this? This is actually like a pointed out, this is system of linear equations. So, notice H times r transpose is actually, what is the dimension of H times r transpose? See H is n minus k cross n r transposes n cross 1. So, H times r transposes n minus k cross 1 and given r, you can eminently compute this vector at the received, very efficiently you can do it, it is not a big problem. So, H n r simply do H times r transpose, it will give you the, it will give you a vector n.

So, this vector is what is call the syndrome, usually it is denoted as, does this centre. So, what you have here like this pointed out is a system of linear equations, how many equations do you have? n minus k equations. How many variables do you have? n variables. So, the variables are here so, this is actually a system of linear equations, have n minus k equations and n variables. So, clearly you can argue the rank is full that there will be 2 power k solutions and it is consistent with what we had before.

So, it is exactly the same. So, the set of all r plus u so, set of all r plus u, this is also size 2 power k, right, it all matches up. So, you will get the same picture various ways, alright so, alright. So, I have, I have a description for the possible error vectors, but now how do I use the knowledge that is biased? So, the question is how do I solve this error, this equation, right.

(Refer Slide Time: 09:36)



So, essential idea and decoding is now solving so, decoding is the same as solving H times e transpose is s. What can I do when I get a particular r? When you get a particular r is the receiver, you can compute s equals H times r transpose and then you can solve for H times e transpose s. You will get 2 power k solutions, which solution will you pick?

Student: Least possible.

Solution with the least possible having weight, why is that? That is the definition of decoding, right. So, then you pick the so, you get 2 power k solutions. So, you pick, oh my God it is becoming a, have a large box so, let us move the box, pick lowest weight solution. And that will be a e hat and then what will be a c hat?

Student: r plus e.

r plus e yes, I will just leave it like this, you know how to make c hat from here, if you want you can do this, add these two and you will get c hat. So, this how a syndrome decoder will look so, it is optimal, it is clear that it is optimal and I am not doing anything so, optimal here. So, this step is e c, the first step is e c computing the syndrome. So, computing the syndrome is e c and then this is called the syndrome equation, you can call this syndrome equation if you like, right. So, the crucial problem is solving the syndrome equation to get the least weight solution, I will get all possible solutions that will be 2 power k, I will be save, I will be stuck in the same old bold, if we do all 2 power k. I want somehow solve smartly for the least weight solution directly.

So, can I do that smartly is the question so, that is what gives you all these sub optimal decoders. So, they to smartly solve this equation for the least weight solution directly, alright. So, this part is the tricky part, it is usually difficult, there are various different solutions for it, but if you want to be optimal, you might want to solve for the exact least weight solution, that is when you will be optimal. So, you do that this is the same as optimal decoding, but I made this comment about 2 power n minus k, where 2 power n minus k come about?

Student: The code word, I mean syndrome that fall in the code word sector, we should not count the e's that belong to c.

Well that is not really, I mean if syndrome is 0 then e will, e might belong to c, see that is not the point, the point is if you make a table here, how big will the table will have to be? That is the question, right, if, ultimately I am thinking of a big table, right for implementing this. So, I have s coming out here, I have to produce e hat, how big, how big will the table be? It will have two columns, what will it have in one column and what will it have in the other? S, that is the important thing, it will have all possible syndromes in one column and corresponding to each syndrome, it will have an e hat. How many possibilities do I have here? 2 power and minus k possibilities. Remember, S is simply n minus k bits. So, I have only 2 power n minus k possibilities for the number of syndromes, each possible syndrome I go through, I get this error vectors.

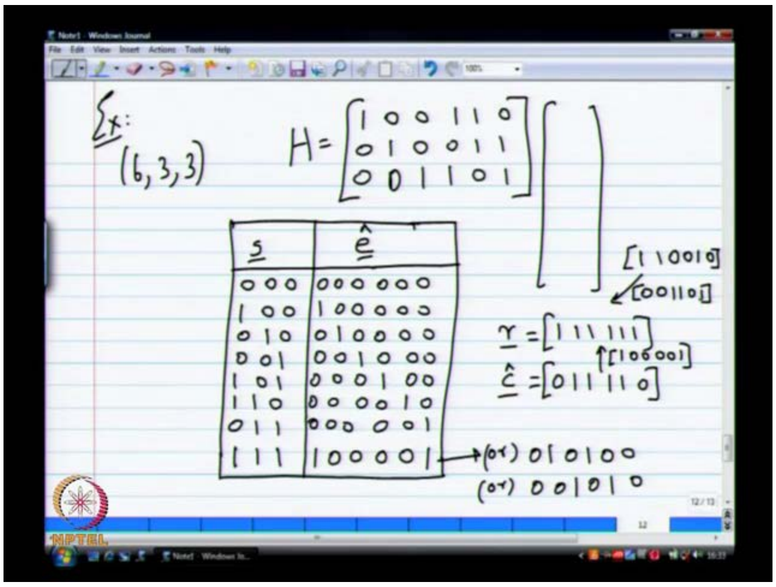
So, the complexity is not as high as 2 power n for instance, it becomes 2 power n minus k. If you want a simply implement then the entire table, you may not want to implement the table, if you want the smartly solve for the lowest weight solution, you can also do that. So, that is why the sub optimal equation, sub optimal decoders come in, like I said if n is 1000, k is 500 you still have a huge table to deal with here. So, if you smartly want to solve for it, you can do that also, that is the sub optimal decoder, but if you want to be optimal you need a table of size only 2 power n minus k, suppose 2 power n in the other case. So, that is where the simplification comes is that okay? Alright?

So, how do you fill up this table? Technically filling up, each row is as complex as what you had before, you do not, you do not gain anything. So, suppose I look at the particular

S, what should I do to find out this, this e hat? I have to solve for H e transpose equals s and that is 2 power k solutions, seems like it is as it very complex, but I should not be doing it, yeah. So, I have to do something else, reason why I have to do something else is I know my e hat is biased. I have to do this operation and reverse, instead of saying given an s, I will try to find an e hat, list out all the most probably hats and find the s corresponding to it that is very easy.

As you keep listing out all the possible e hats, see if you are covering all the possible s's. Once you have covered all the possible s, you can stop at that point. So, that is a kind of reverse way of building this table and that is usually more efficient than trying to solve all of them and every step you will get something, not like for each row you do not have to spend 2 power k. So, over all complexity remains at 2 power n minus k, that is what I am trying to say, alright. So, I have described this from my, an abstract way so, we have to see examples and when we see examples, it will be a little bit more clear.

(Refer Slide Time: 16:18)



Ex:

(6, 3, 3)

$$H = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} \\ \\ \\ \end{bmatrix}$$

$\underline{s}$	$\underline{\hat{e}}$
000	000000
100	100000
010	010000
001	001000
101	000100
110	000010
011	000001
111	100001 $\rightarrow$ (or) 010100
	(or) 001010

[110010]

[001101]

$\underline{r} = [111111]$

[100001]

$\underline{\hat{c}} = [011110]$

So, let us go back to this example that we had, hopefully I will be able to reproduce that same example, the 6 3 3 example. So, here is the 6 3 3 example, for this when we try to do this by hand we saw that building the table for the, in the previous case needed, 64 entries for all possible r, 2 power 6 that 64, it is a lot of entries. But you will see if you do syndrome decoding, the syndrome table, by the way this is call the syndrome table, I will name it, let us call the syndrome table. Building the syndrome table on the other hand is not too hard, at least for these cases when you do it by hand, it is quite easy.

So, let us see, let us try and build the syndrome table like I suggested. So, you have the syndrome s and the e hat. In fact it is so easy you can do it either way, you can exhaust all possibly hats or you can go syndrome by syndrome and try to find out, what is the least weight e hat? Both of them you can try, but the first thing you should write down is what?

Student: All 0.

All 0 e hat, right; because that is the most the trouble one, and that will give you the syndrome 0, right. The next things you should try are error vectors of weight 1. So, you try them for instance you do 1 0 0 0 0 0, you will get 1 0 0 here, you do, it is some piece of laborious writing down, we can do that 0 1 1. So, how many syndromes have I covered? Seven different syndromes have already been covered, only one syndrome remains, alright. So, you have to keep looking for weight two error vectors, right. We have exhausted all the weight one error vectors, I have got only seven syndromes.

Now I have to look for weight two error vectors, now you can be a little bit smart, you can do this very easily. So, you know the missing syndrome is what? 1 1 1 I want a weight two error vector which will give me h e hat equals 1 1 1. So, how do I do that, imagine here I am putting a vector here with 2 1's and when I multiply h times e hat, I should get 1 1 1. So, how do I do that?

Student: Second and fourth.

Second and fourth for instance or first and last or so, now you are noticing an interesting thing there are more than one possibility even for the least weight. How do you, what will you do if you do that, if you have that, you can pick any one without losing anything in probability of error. So, what this happening is you are receiving a, receive that r it is equal distance from multiple code words. So, it is happening for some crazy reasons it can happen, you will end up and you can go back to any one and you would not lose anything in probability of error, you can go back and see the competition, you will see probability of error will remain the same. So, will be optimal whatever choice you make here.

So, let us do this choice for instance so, this here you can put or 0 1 0 1 0 0 or what? 0 0 1 0 1 0, any one of these three e hats you can put there, is that okay? So, there are lot of useful notions that come out of thinking about the decoder as a syndrome decoder as appose to searching for r to c, okay go ahead there is a question.

Student: How do you know that always get all possible values for S from the h r transpose?

Well if it is a full rank you will get it, eventually you will get, if h is full rank, the question was, how do know that if I vary the e here I will get all possible 2 power n minus k, if h is full rank you have to get it. Because, there will be some columns which are linearly independent, you put all possible vectors there, eventually you will get it. So, you should get all possible ((Refer Time: 21:41)). So, there are lot of notions in this syndrome decoders so, let us go back and look at this block diagram. So, this, remember this is the block diagram, I have that table you have to use it in this block diagram, given an r, what should you do first? Compute s equals H r transpose and then go to the table, look up the e hat and then x r with the received r and you get c hat.

So, this notion of figuring out the error really gives you this notion with your correcting for an error vector, right. There is an error vector that is introduced by the channel, I am trying to figure out based on the redundancy I have added in the code words, based on the gap I have between the code words. What was the error introduced by the channel and i am cancelling. So, you are literarily correcting for the error introduced by the channel, it gives you the nice, I do not know, it is in cute as way of thinking about it, it gives you that. From there you can make a list of what is call the correctable error vectors.

So, if I have a syndrome decoder, if we have a syndrome decoder with a certain syndrome table, whatever it is, the error that was introduced by the channel, the errors that I am declaring at the decoder will always be from the, from the table only. I will always say this e hat will come from the table. So, this error vectors in the table are called the correctable error vectors, any error vector out size of this table is not correctable, that is also an interesting statement. So, let me go back to the syndrome table. So, maybe I should spend little bit more time with this correctable, in correctable with this example before I go out of this syndrome table.

So, suppose in the actual transmission the error, the r that was received was now, in case you remember the previous thing that I gave you. So, r that I received was let us say 1 1 1 1 1 1, what will be c hat? You can now answer this question without too much, too much of, without too much work, right can we answer this question? Let me see.

Student: ((Refer Time: 24:07))

The first thing you compute is what? The syndrome, what is the syndrome corresponding to this r? So, you have to do h times r transpose. So when you do that, what happens? You get 1 1 1, you can quickly do that, you will get 1 1 1 then you go to this table, look up the e hat corresponding to that and then x r it with the r and you get c hat to be 0 1 1 1 1 0. Now this would have been very hard, if you did the exhaust to search over all possible code words. So, you see works out quite easy this simply picks out the last gain. Now comes challenging question, I want to ask challenging question, can give me any other r, give me any other r which will give you the same, same row, give one other r which will give me this 1 1 1 0 0 0, right.

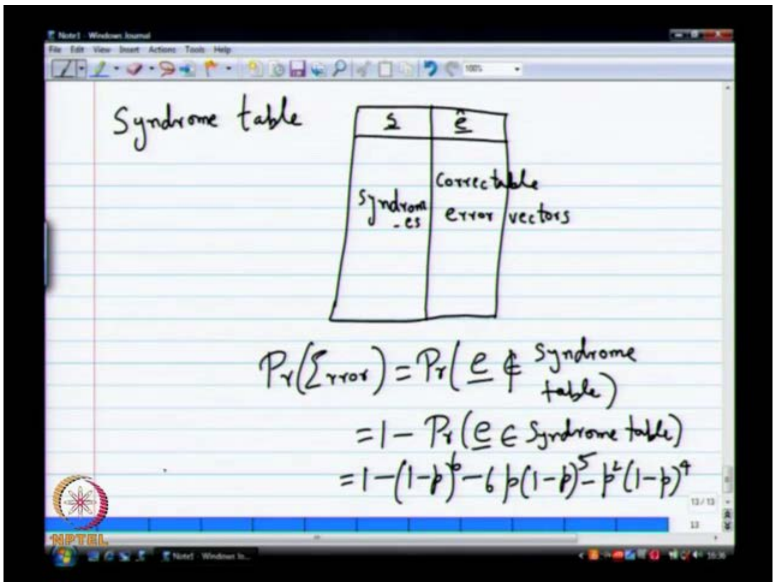
So, what is, will it, will come up any, any number of r how many other r will that be? 2 power k so, it will be exactly 8, fine 8 other code words are the same vectors, which will give you the same syndrome, alright, that is one thing. The other thing is see, right now I am claiming my e hat was 1 0 0 0 0 1. So, when I declare this, what I am saying? I am saying this r was received when this, this code word was transmitted and the error vector 1 0 0 0 0 1 was added to it, right. What is equal possible is, you can have another code word, let us say for instance I do not know any other code word plus any other error vector, some other error vector, not any other, some other error vector which will give you the same r, it is possible or not?

For instance this, just a silly case you just pick the code word to be all 0, then what will be the error vector? All 1's, you will get the r, alright. So, we are only correcting errors with some probability, we are not saying for sure that this was the transmitted c hat, you can never say, if you can say then there is no decoding problem, right. Some errors have happened, you are doing syndrome decoding and you are saying e hat was this, then c hat was this, that is correct only with some probability. But that probability is maximum, you know its maximum, but that could be some other code word, for instance I mean just a trivial case to take 0 0 0, but you can any other code word.

Let us say for instance 1 1 1, let us say 1 1 I should not take, I should not start with 1 1, may be 1 1 0 0 1 0 and then 0 0 1 1 0 1. This could be another error vector that got added, but notice here the probability of this error vector is only, is what? It is p square into 1 minus p power 4, what is a probability of that error vector? p power 3 into 1 minus p power 3, that will be definitely smaller than the error vector that I picked, you can be sure that you are picking the most probable error vector. But there are definitely other code words and other vectors with non-zero probability, which could have given you the same r. So, you can make errors even with the syndrome decoder, do not be under the assumption that you can never make errors, you will make errors.

So, that is why I have this notion of a correctable error vector is important, when, no matter what error vector actually happened at the decoder, you always declare that the errors that happened was those errors, that are in this table. You will never declare that the error that happened is outside this table, right. If some error outside this table occur, you can never correct it, though it occurs in the lower probability, but if it occurs you cannot correct it. So, that is the idea.

(Refer Slide Time: 28:36)



Syndrome table

$\underline{s}$	$\hat{\underline{e}}$
Syndrom-es	Correctable error vectors

$$\Pr(\text{Error}) = \Pr(\underline{e} \notin \text{Syndrome table})$$
$$= 1 - \Pr(\underline{e} \in \text{Syndrome table})$$
$$= 1 - (1-p)^6 - 6p(1-p)^5 - p^2(1-p)^4$$

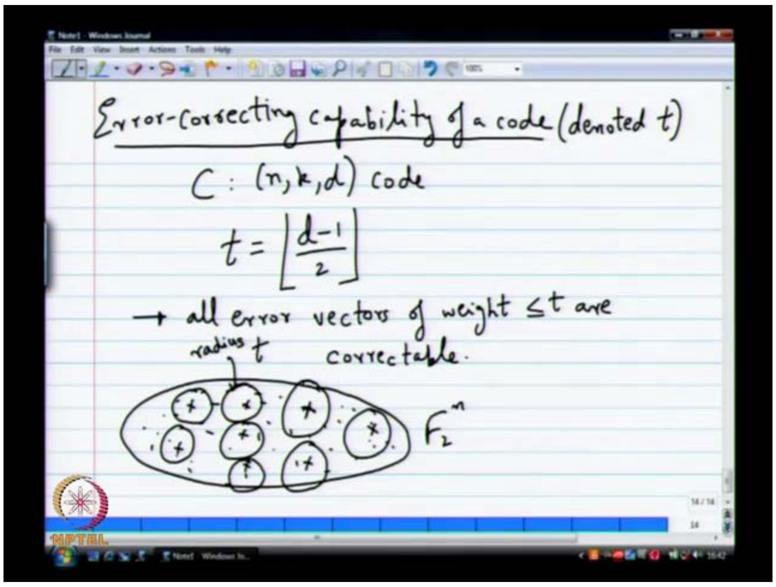
So, in a syndrome table, what you have, the e hats you have are the so, here you have all syndromes, what you have here are the correctable error vectors. So, what is easy to compute, once you have this syndrome table is the probability of correct decision or probability of error. What is a probability of error? Probability that, the e that occurred does not belong to the syndrome table, you can compute it as 1 minus probability that e belongs to the syndrome table, this is very easy to compute. Why is it easy for in since, in this example what is the probability e belongs to the syndrome table? You have to simply compute the probability of all 0, 1 0 0 0 1 0 on, so on and then add it up.

So, what is the probability of all 0 error vector? 1 minus p power 6, what is the probability of remaining vectors? This guy is p into 1 minus p power 5, how many of them do we have? 6 of them do we have, 6 of them we have. So, it is a 6 p times 1 minus p raise to the power 5, then what about the last guy? p square into 1 minus p raise to the power 4. So, these are, these are interesting exercise problems and they are also interesting exam questions. So, given parity check matrix come up with the syndrome table and then find the probability of error.

So, remember this is the lowest probability of error that you can ever get for this code, for the 6 3 3 code that we had, this is the lowest probability of error you can possibly get, you can never get better than that over a b s c of p and how do you compute that? This is a question. So, somebody ask you given a code what is the lowest probability of error that you can find? How do you go about doing it? Find the syndrome table, find the correctable error vectors, simply 1 minus probability that the error is in the syndrome table, you can get. Simple step one to step procedure, it can be.

So, I did just, I just did one example so, hopefully you can see quite easily how this example can be generalised. Any other parity check matrix I have, you just repeat the same problem, same thing one after the other you can do and in a tutorial problems I do ask some several questions like this. So, you will see and you practise some of these things. So, it is not a big, that is a not a big deal. So, one more property that you can now define even this notion of correctable error vectors is, how many errors can you correct, that is the next notion, error correcting capability of a code.

(Refer Slide Time: 32:38)



Error-correcting capability of a code (denoted t)

$C : (n,k,d)$ code

$$t = \left\lfloor \frac{d-1}{2} \right\rfloor$$

→ all error vectors of weight $\leq t$ are correctable.

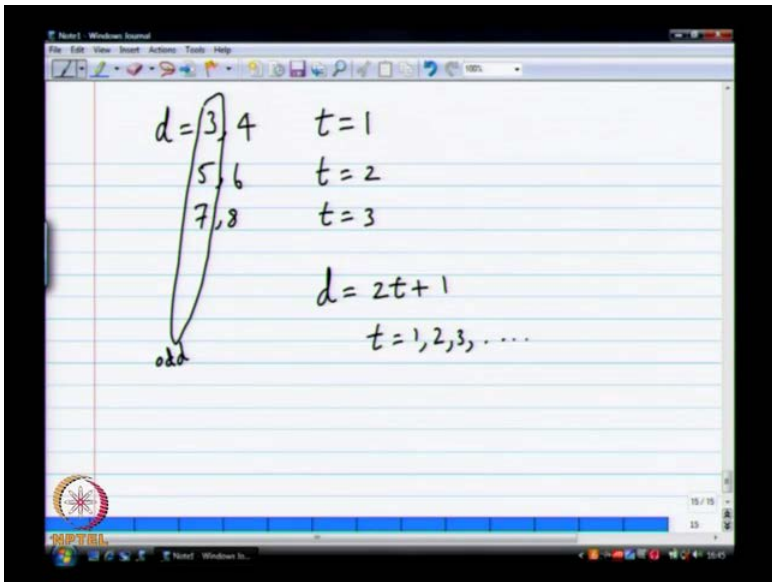
So, that has a lot of definitions. So, suppose you have an n k d code this is usually denoted t, one can show that the error correcting capability t is d minus 1 by 2, I have not even decide, defined it, I will tell you what I mean by it. When I say error correcting code capability is t, it means mean what, what does it mean? All error vectors of weight less than or equal to t are correctable. So, it is very easy to define this once you know what it is. So, what is the set of correctable error vectors? Those error vectors that are in the syndrome table, all error vectors weight less nor equal to t are correctable implies t is the error correcting capability. It is well defined, you do not have to worry about not being well defined, it has being very well defined quantity so, you can define it very precisely.

So, let us go back to this example, if you go back to this example, sorry. So, you have 6 comma 3 comma 3 code, from the computation you see t equals 1, right 3 minus 1 by 2, you get 1. Clearly in the syndrome table you have all error vectors of weight 1 appearing, it is not 2 because there is only one error vector of weight 2 that is appearing here, there are several more of them that are not appear. So, that is the idea of error correcting capability, there are also useful way of visualizing error correcting capability, which is once again interesting.

So, once again we will do this picture, which represents f to n, all these dots which are n bit vectors and what are the stars? Stars are the code words. So, you have transmitted a particular code word, when you add errors to it what is happening? You are moving away from the code word, if the error was of weight 0 then you will be at the code word itself. The error was of weight 1, you will be within the hamming ball of radius 1, the error was a weight 2 you will be within the hamming ball of radius 2 etcetera etcetera. Now if you draw hamming balls of radius t, what will happen? They will not cause any intersection, there will be no intersection for hamming balls of radius t, the t is d minus 1 by 2. So, this is radius t.

So, we saw this idea before when the, in the hamming bound a very similar ideas happening here. So, if at all only t or few errors are introduced, you should be definitely able to correct them because the minimum distance will be to a unique code word. There will be no confusion, you will never go further away from the transmitted code word, you will be able to correct them, there is no problem that another interpretation for this t. So, that is the error correcting capability of a code. Any questions? Everybody totally exhausted, still have 38 minutes to go so, a long way a head, alright.

(Refer Slide Time: 37:01)



$d = 3, 4 \quad t=1$

$5, 6 \quad t=2$

$7, 8 \quad t=3$

odd

$d = 2t+1$

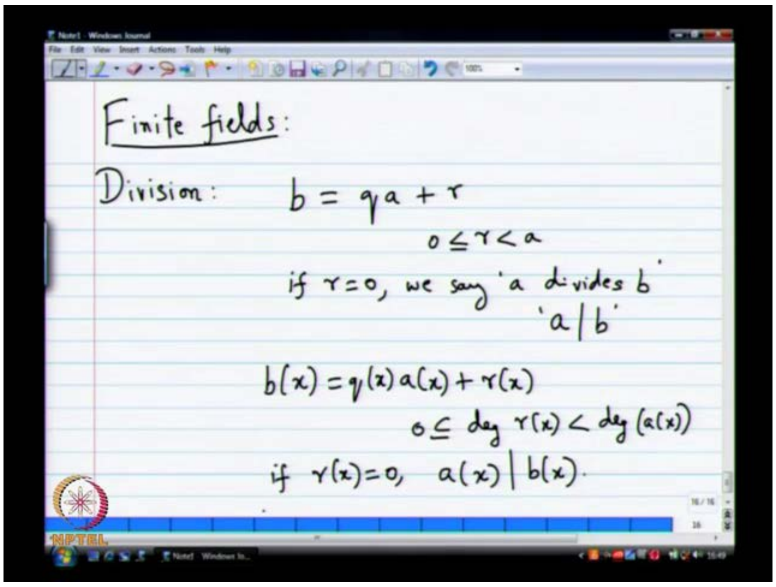
$t = 1, 2, 3, \ldots$

So, now you see why d equal 3 and d equal 4, are not really making up significant difference. If you have both d equal 3 and d equal 4, what happens? Your t is still 1, right and this will continue if you have is 5 and 6, then t becomes 2, you have 7 and 8, t equals 3. So, usually that is way people will worry about d of, d that is odd only. So, in fact usually its very common to take d to b 2 t plus 1 and then have t as 1, 2, 3 etcetera. So, this is what you usually want so, this will give you a t error correcting code, right. From r design so far, we have only one other correcting codes, we are not able to design a two error correcting codes.

So, we have to go d equals 5 for that and you will see it is a bit non-trivial finally, aloneness it is not so easy, ultimately you can do it, it is not impossible. But it is a little bit more difficult and complicated, once you go to non-binary fields you will see construction becomes much easier. So, that is the reason why people move to other fields, think of other interesting constructions. So, I think, this kind of brings us to the end of binary codes and all that I wanted to say, we are going to move towards finite fields. So, in case there are any questions comments or anything that is disturbing you, it is a good time to ask.

If you all are reloaded, all your batteries recharged. So, I think this is a good place to stop as far as binary codes are concerned. So, if you want quick recap it is binary codes are k dimensional sub spaces of f to n, you have to generate a matrix which describes the binary codes, you also have the parity check matrix. When you have minimum distance, which is a crucial property which controls as the correcting capability, minimum distance is connected to the parity check matrix. In fact design parity check matrix with several minimum distance, then the parity check matrix also plays a key role in the decode. It gives you a nice implementation for the decoder, it gives a feel for how the decoders work etcetera, etcetera. So, that is where we are going to stop and move towards finite fields.

(Refer Slide Time: 40:06)



Finite fields:

Division: $b = qa + r$

$0 \leq r < a$

if $r=0$, we say 'a divides b'

'$a \mid b$'

$b(x) = q(x)a(x) + r(x)$

$0 \leq \deg r(x) < \deg(a(x))$

if $r(x) = 0$, $a(x) \mid b(x)$.

So, before I move to finite fields couple of, couple of quick, quick, quick back ground material so to speak I mean, it is not really, I mean I would not do, I would not do all the background, I will assume some of the back ground you know already. But something which you might not have, might not have forgotten, you may not have learnt it in the capful way with some times it can come back to you. So, one of the things I see people I confused about is usually division. So, how do you describe division? So, division with reminder is what I am talking about. So, if you have two numbers a and b, when I divide b by a, what am I doing?

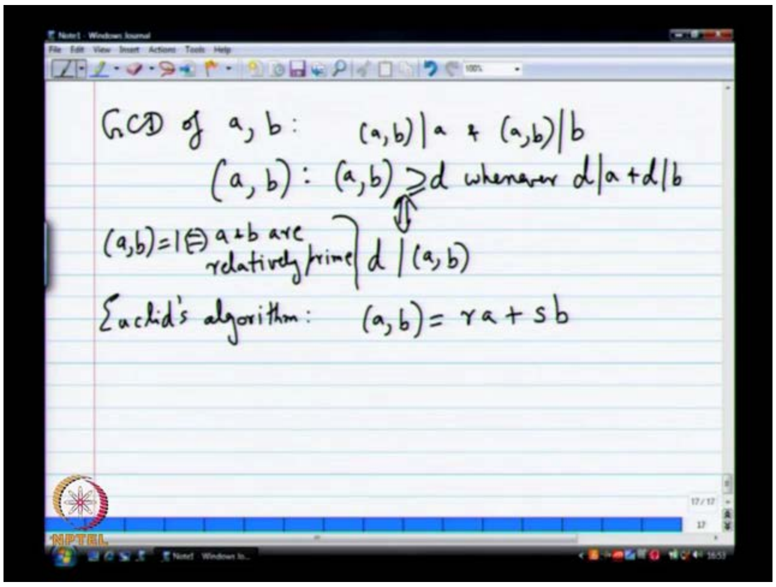
So, many descriptions come out. So, the way, the way that, the way that is most useful at least in this course is to think of division of b by a as finding two other number q and r such that, b equals a q plus r and then r is between 0 and a minus 1. So, that is the crucial way we should think about division. So, division with reminder when you divide b by a, you are basically finding the quotient and the remainder, p equals q a plus r, usually q may all these are integers. So, I have not said anything about what these thing are, everything is an integer, we are trying to find an integer q and r and there is a restriction on r, r as to be between 0 and a minus 1, strictly less than a and can be 0 also, goodness saying, sorry.

Student: Your palm is touching sir.

How can you write without your palm touching? Like this, I am not going to write like this, things are open I think that is self opens. Yeah, I think hiding the task bar is the good idea, that is something I should try later, alright. So, this is, this the way to think about division so, when do you say a divides b, if r is 0 we say a divides b, divides b. So, this is usually written as this notation, this is the same as this notation. So, this is just short hand for a divides b. so, there is a similar division for polymer also, it is a little bit more involved, if it is also this very similar, but little bit different. So, if you have two polynomials b of x and a of x, when it divide a of, b of x by a of x, what are you actually doing?

You are finding two other polynomials, one is call the quotient polynomial q of x, other is called the remainder polynomial r of x and what is the condition that is true there? Yeah, degree of r of x is degree of a of x. So, that is side a so, once again one you say a of x divides b of x, r has to be equal to 0. So, once again the same condition true, it is true, alright. So, next there are, there is one more quantity called the GCD, the greatest common divider, we may not use it too much, just letting you know because sometimes I might needed somewhere and then I should be lose somewhere. GCD is basically the greatest common divisor of a and b.

(Refer Slide Time: 44:18)



GCD of a, b: $(a,b) \mid a$ & $(a,b) \mid b$

(a,b) : $(a,b) \geq d$ whenever $d \mid a$ & $d \mid b$

$\Updownarrow$

$(a,b)=1 \Leftrightarrow$ a & b are relatively prime $\quad d \mid (a,b)$

Euclid's algorithm: $(a,b) = ra + sb$

So, how do you design, define GCD of a and b? It is usually denoted like this, this GCD of a and b is what? It is such that a comma b is what?

Student: a x plus b, integer divides both a and b, d such that no other, it is a greatest number.

So, it is such that, such that a comma b is what? Let say greater than d whenever, whenever what? d divides a and d divides b, also then what should happen? a comma b should divide a and a comma b should divide b. In fact, see this is the definition of the greatest common devise, devisor right. So, it is a devisor of both a and b, common devisor and then it the greatest such common devisor, if you have any other common devisor, this is going to be greater than that. In fact you can use this definition and the division property, division property as in division algorithm that exist in the integers to show that in fact this is true, what is true? It will also divide, no I think it is rather way round, d will divide a comma b, but any way that does not, it is not truth to crucial here.

But this, these are equivalent to d dividing, I am sorry, if you want I will put k here. So, it is d dividing a comma a comma b so, think about how you might want to prove it, in case if you want it simple problem in this one and such things. Also another very interesting property, which is called, where it is very famous algorithm for finding GCD you can use that algorithm to also do that, it is Euclid’s algorithm. Euclid’s algorithm gives you a way to do the following, it turns out the GCD a comma b can be written as half times a plus s times b, where r n s are also integers. So, there is a linear combination of a and b which will give you the GCD of a and b, this is also an interesting results. You do Euclid’s algorithm, for instance how do you find GCD?

You start with b, divided take the remainder then push a n r as the devisor and the dividend, then you can proceed like that. So, you just undo the Euclid’s algorithm, you will get this r n s, it is not easy. So, at every stage you only doing linear combinations with the questions, it is very easy to see that so, you get this formal. So, in particular if a and b are relatively prime, when do you say a and b are relatively prime? If a b equal to 1, then you say a and b are relatively prime, what is relatively prime mean? They have no common factors. So, what should happen if a and b are relatively prime? There should be a linear combination of a and b, which gives you 1 so, that is an interesting property.

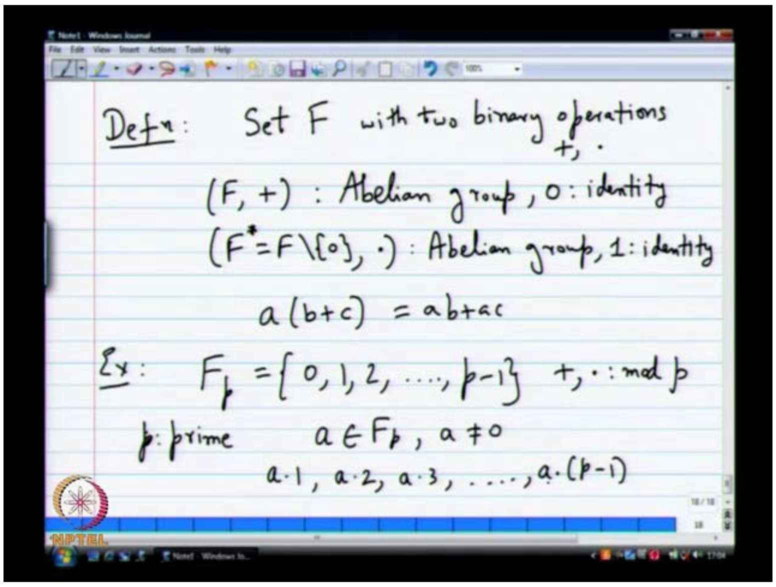
So, all these things are true also for polynomials, if a of x and b of x are polynomials with coefficients from some fields. So, they have to have coefficients from some fields, otherwise nothing is true, co efficient should come from some field usually like real numbers or something like that or rational numbers. Then all this things are true, you have also GCD for ration, polynomials then you also have notion of polynomials being relatively prime. Then you also have the notion of this GCD being able to be when you can write it as linear combination of a and b.

So, you know I am saying this all real quick we will not really use all of this. But in case you have to, the reason why I am giving you this is, in case you have to prove some of these properties, you will have to use these results, some other properties that I am going to write done, may be you might have to use these results. And they will quickly point to the proof, I would not do many of the poofs in detail. But I will quickly point to the proof and I will mention some point as like this, say you use these properties and you can prove it just. So that, we are on the same page we have, I have to write this term. Also, some of the formulas that I want, I want to write will involve things like GCD.

So, it is good to know what the GCD is and how it does related to the numbers. So, any of this is really troubling you, do not worry about it too much, when we actually come to it, I will explain it more detail. But you would not need to know much more beyond this, one way of finding GCD is what you just said, do the prime factorization and then pick out the largest power, smallest power of all the common prime factors. I can give you the GCD, that is all we find it prime factorization is not easy, right.

So, Euclid's algorithm is an easy way of finding GCD, can be found very easily, the, you see Euclid's algorithm. How many of you remember Euclid's algorithm at all? No, maybe little bit and I am sure you will start to remain in school, but just not that familiar. So, these are somethings that will be needed. So, let us just jump into the definition as well, when we needed we will may be come back to this or may be not, alright.

(Refer Slide Time: 49:53)



Defn: Set F with two binary operations $+, \cdot$

$(F, +)$: Abelian group, 0: identity

$(F^* = F \setminus \{0\}, \cdot)$: Abelian group, 1: identity

$a(b+c) = ab+ac$

Ex: $F_p = \{0, 1, 2, \ldots, p-1\}$ $+, \cdot$: mod p

p: prime $a \in F_p$, $a \neq 0$

$a \cdot 1, a \cdot 2, a \cdot 3, \ldots, a \cdot (p-1)$

So, we are going to see finite fields, I wrote down the definition for fields before, I want to quickly remained you what the definition is, the definition of the field. So, you have a set F with two binary operations, but once again, why are we going to finite fields? What was the logical progression that took me to finite fields? Yeah, we want to design codes will be equals 5 and higher and the only construction the people have so far, which is nice to describe is uses this fields ideas. So, that is one of the reasons we are going, but even otherwise finite fields have so many other applications and so many related areas, it is good for you to know them.

So, you do not think that these are very abstract object used only in coding theory, used everywhere, only when things not, two binary operation which we will call plus and dot. Plus we will call as addition, dot we will call as multiplication, we want the following conditions, right. I will write only sketch, I think because the last time I gave you a very precise definition. So, quickly writing it f comma of plus should be a Abelian group, f star, what is f star? So, Abelian group means clearly you should have identity elements. So, 0 we will take as identity of, additive identity.

So, it is just a notation for the additive identity in f, we will call it a 0, f star which is f minus 0 comma dot should also be a Abelian group. And one will be termed as the identity element of the multiplicative group and there are some, it is some conditions I mean for instance, multiplication has to distribute over addition. So, such things should be true, a dot b plus c should be the same as a b plus a c etcetera, etcetera. So, all these are the conditions you can put, I think this is the only other conditions, I think is anything else, we put etcetera, etcetera maybe I will remove it, this is also, this should be there.

So, multiplication this should distribute over addition this, this gives you a field. So, basically from very high level field is something where you can, where you have a set of elements and you can add to elements, you can also subtract them in a group, you can add and subtract right or you can multiply and divide. So, in the, in the field you should be able to do all four, you should be able to add, subtract, multiply, divide, divide by 0 is not needed. So, it just extends your notions of field to something more abstract. So, it has some elements, if you want to be very definite you can think of the rational field, the rational numbers are the nice field that you can imagine in your head or you can think of the binary field f 2.

So, one of the non-trivial examples which is also finite is this f p, p is prime, what is f p 0, 1, 2 so on till p minus 1, addition and multiplication are basically integer addition and integer multiplication, modular p. So, basic if you, if you let the size of f be infinite, you can get a wide variety of fields, all kinds of fields are possible. So, this is lots of interesting stuff there also, but if you restrict f to have a finite number of elements, the fact that it needs to have a finite number of elements. And it has to be closed under two different operations, that interact with each other with distribute of sense is a serious limiting thing. You do not have too many finite fields, finite fields have a very precise structure.

So, we will see it soon enough, but f p is once such example. So, yes once such example and you see there are too many conditions, right, you have to be able to add them and still you should not leave the field and every element should have a inverse also, hopes in the same field. All these things are fairly serious limitation, does not seen very obvious when how do you, how to pick out fields etcetera. So, this for instance is a field, what are the various properties that are easy to verify? If you want to verify some of these properties, which one is easy to verify? f comma plus being a Abelian group is a very easy thing to verify.

So, you add any two closer is definitely there, inverse, how do you find inverse? If you have any I, what is the inverse of that? You explicitly write down the auditor inverse, no problem. So, Abelian group for the plus additive group is easy, what is lightly non trivial is f star with dot modular of p being an Abelian group, it is a little bit non trivial. Once again closer is not a problem, identity is not a problem, inverse is the only thing which is bit a sticking issue, you can directly see that every element has the multiplicative inverse.

So, it is not very obvious may be there is obvious to some, but it is not very obvious. There are some non-trivial numbers their results very simple ones, which we will see in the first chapter of every number theory book, which will give you the fact that everything has a multiplicative inverse also. But if you have not studied it, it seems quite fantastic that for any prime number, right, any other number, any other number a less than the prime number, I have an a inverse which one multiplied modular p will give me 1. So, that not very obvious, it is a difficult thing to prove, for instance of 5, right if I take p equals 5, what is the inverse of 2, multiplicative inverse of 2, 3 right.

Suppose I take a large prime, give me a large prime I do not know it is difficult to find, let us see 31, it is not that large it is prime, what to say inverse of 25, it is not easy right, not easy. So, it is just becomes a little bit more difficult to visualise, but the factor that it exist can be shown quite easily may be we will see a quick proof, we will come to that later. The distributive property is also very easy, it just comes from integer addition and integer multiplication, right. Integer addition and integer multiplication has the distributive property. So, when you do the same thing modular p, you also have the interior, integer property, there is no problem with that, modular does not change anything with the addition and multiplication, you can do all kinds of steps also, no problem.

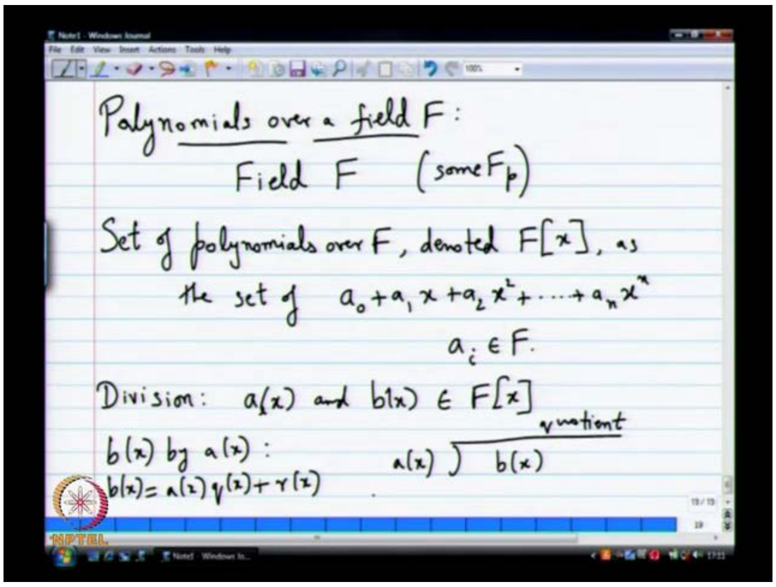
So, the only non-trivial thing is to show what? Yeah, so that is the only thing I want to show. So, let me see there is a quick proof here, which I can easily repeat for you. So, here is a simply enough proof which we think is, which I think will work. Suppose I take some element a, suppose you take a in f p and a is non-zero, it is important. Now look at this following list of elements, if list of elements I am going to look at that are a into 1, a dot 2, a dot 3 so until a dot p minus 1, when I say dot what do I mean? It is multiplication modular p.

So, my claim is there is no repetition in this list, remember there is closure so, all of these things belong to what? f p star from 1 to p minus 1, we would not get anything else. But my claim is there is no reputation in this, once I show there is no reputation in this, what has to be true? One of them should be equal to 1, that shows my inverse, but how do I prove that there is no reputation in this? Yeah, you can show that for instance if a times i equals a times j mod p, what should happen, a times i minus j or j minus i depending on what is larger should be 0 mod p, it means p has two factors. But p is prime, cannot have a 5, is that a good proof, everybody is happy, think about it.

So, it is, it is an interesting proof so, that is how you prove it. So, notice the proof is not constructive, it is not explicit does not tell you for a what is the multiplicative inverse. In fact there is an explicit way of writing it down also but any easy it do not matter so, it does not give you, it does not give you what this explicitly, but implicitly it says it should exist. I know I did not write down all the steps, I do not mean to write down all the steps, you please fill in if you feel like it. So, it is up to you it is not important to know all the steps of the proof at least in coding, you should know how the proof is built, is that okay? Any questions on this proof, no alright.

So, it would, would not work if p is not prime, you see that p is not prime, you can eminently have a times a minus i is equal to 0 mod m, there is no problem. If p is prime it works, is that okay, alright. So, this is a finite field so, you have one finite field of course, there are, there are, you might say there are infinite number of prime. So, there should be an infinite number of finite fields, yeah infinite number, but quite few know number of primes between 0 and 1, n are like log n, so it quite a few, very few number of primes in the, in the, in the list, alright. So, what else do I have to say?

(Refer Slide Time: 01:00:25)



Polynomials over a field F:

Field F (some F_p)

Set of polynomials over F, denoted F[x], as the set of $a_0 + a_1 x + a_2 x^2 + \dots + a_n x^n$

$a_i \in F$.

Division: a(x) and b(x) $\in$ F[x]

b(x) by a(x):

$$a(x) \overset{\text{quotient}}{\overline{\big) \; b(x)}}$$

$b(x) = a(x) q(x) + r(x)$

So, the next thing that is the little bit interesting is k polynomials over a field, I want to, I want to say few more things formally about these things, then we will proceed. So, if you have a field F, think of it as some finite field for now, think of it as some F p. We will define, first let us define, we will define the set of polynomials over f, which is denoted F with square bracket x as the set of basically polynomials. So, what is polynomial? a 0 plus a 1 x, x is the variable a to x square plus so on till some a n x power n, only a finite number of terms. So, n is some finite number this a i is belong to the field f. So, these are also call polynomials with co efficiencies from f, all of you knew this right so, no problem, polynomials of finite number of terms of this form a 0 plus a 1 x plus a 2 x square so on.

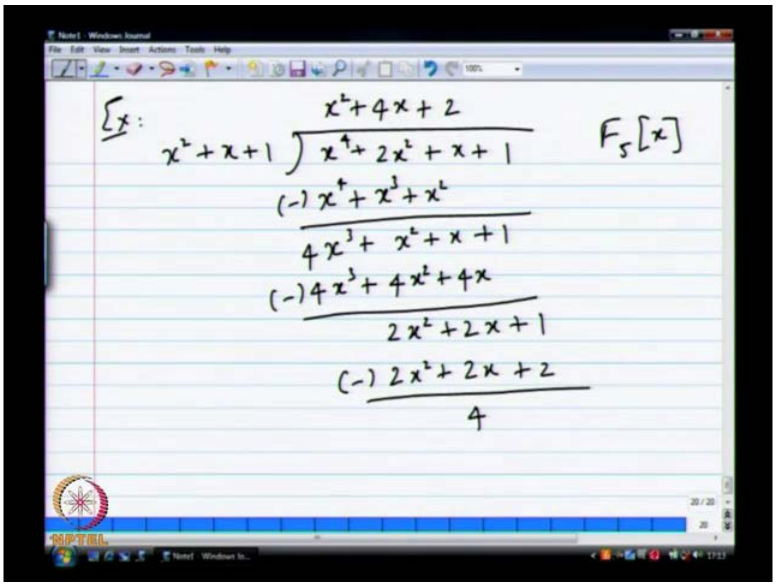
So, of course what I wrote down about division is very very important so, if you have division is the most crucial property for polynomials, which we will use quite extensively. So, if you have two polynomials a of x and b of x belonging to f x, there should come from the same f x otherwise, division will not make any sense. How will you do polynomial division? You know how, you remember the long division method, right so, do not tell me you forget that. I am not going to do the long division method for you, how do you do the long division method for polynomial?

So, you need to do b of x divide b of x, let us say divide b of x by a of x, can you do that, what do you do? You put b of x inside the, you write, you write something like this, you must remember this, you put b of x here, everybody is laughing I do not know, but I was surprised that some people did not know this, you put b of x a of x and then what do you put here? You put the co efficient needed to cancel the largest degree term, but see you have to remember this coefficients are now coming from the finite field. So, you have to pick the, pick a suitable term so, you have to pick a suitable term which will be the coefficient of b divided by the coefficient of a that you want, the largest coefficient.

So, everything is in the finite field now so, the division, division is a little bit more complicated, it is not as simple as just putting b by a, rational field is what you must be news to and you simply write b by a there, right, you cannot do that here. So, you have to go to the finite field every time you want to divide and can call the first term, but other than that the long division steps are exactly the same as the long division you learnt in school, there is no difference, you do that. Finally, what you have here is the quotient and after lot of steps, what do you have at the n when the degree goes below a is the remainder, when you cannot cancel any more you get the remainder.

So, that is the idea that quotient and that remainder can be written in this form, those are the unique q of and r of x such that it is unique, it is unique degree of r of x is between 0 and degree of a. So, you write that formula right so, b of x is a of x into q of x plus r of x. So, it is the same division, it is nothing really different, it is exacts same division that you have learnt from your school, except that the coefficient are from the field. So, you will have to do division in the, in the field, if it is finite field you have to do division modular that p, for instance if it is f p you have to do division modular of that f p. So, it is a little bit different, division will work different from your intuition so, thus division and then actually have an example worked out, may be you should do that example. It is a good idea to do that example.

(Refer Slide Time: 1:05:29)



Ex:

$$\begin{array}{r l} & x^2+4x+2 \\ x^2+x+1 \,\big)\!\!\! & \overline{x^4+2x^2+x+1} \\ & (-)\, x^4+x^3+x^2 \\ \hline & 4x^3+x^2+x+1 \\ & (-)\, 4x^3+4x^2+4x \\ \hline & 2x^2+2x+1 \\ & (-)\, 2x^2+2x+2 \\ \hline & 4 \end{array}$$

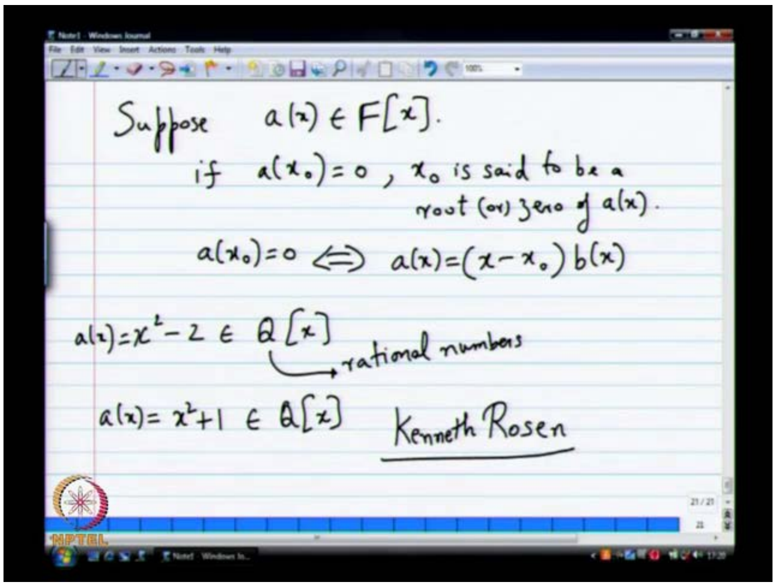
$F_5[x]$

So, here is an example, how you divide for instance x power 4 plus 2 x square plus x plus 1 by x square plus x plus 1. This division let us say is an f 5 of x, I am imagining that these polynomials are in f 5 x, they could also b in f 7 x or f 11 x or anything, I am just saying f 5 was a little bit easier. The first term is easy, why is the first term easy? It is just 1 1 so, we will just put x square for the question, then what you put here below this, x power 4 plus x power 3 plus x square and then you add these two things, when you add these two things what happens? x power 3 remains, I am writing in a little bit nonstandard, I mean usually you leave a gap here, right so that, you can fill in the x power 3, but it is okay. 3 x square plus, I should subtract here, right I am sorry 4 x cube plus x square. So, you have to subtract so, I am still in the binary mode so, I am just adding.

So, when we go to f 5 we have to definitely subtract, you cannot add, you cannot get anything. So, plus 1 right yeah 4 x cube plus x square plus x plus 1, then what do you put here 4 x. So, then you get what here 4 x power 3 plus 4 x square plus 4 x, you do that what do you get? 2 x square plus 2 x plus 1, it that right. This will cancel, 1 minus 4 is 2 in a 5 it is minus 3, minus 3 mod 5 is 2 again 1 minus 4 is minus 3 mod 5 is 2. So, remember this is minus, do not do plus, I think if you doing plus you will get 0 already, do minus and then you put the last term, which is 2 and you get 2 x square plus 2 x plus 2 and you have to again do a minus so, you get 4. So, you can verify this formula, you can verify my formula right b of x is q of x times a of x plus 4.

So, this is the division and I mean other point of doing this example is that, it is not really that hard, it just written in the notation. And all that you get a little bit confused, but you may not have known that result, that b of x is equal to q of x a of x plus r of x, when degree of r is restricted to be between 0 and less than degree of a, you get unique q of x and r of x, it is a good thing to know. So, division is fine and then 0 is fine.

(Refer Slide Time: 1:08:32)



Suppose $a(x) \in F[x]$.

if $a(x_0) = 0$, x_0 is said to be a root (or) zero of $a(x)$.

$a(x_0) = 0 \iff a(x) = (x - x_0)\, b(x)$

$a(x) = x^2 - 2 \in \underbrace{\mathbb{Q}}_{\text{rational numbers}}[x]$

$a(x) = x^2 + 1 \in \mathbb{Q}[x]$ Kenneth Rosen

So, then the next thing is 0's of a polynomial. So, that is also important, what are 0's of f of x. Suppose so, it is very typical to denote as a f of x. Suppose, you have f of x belonging to or let us say, a of x, a of x belonging to f x what do I mean by this? a f x is the polynomial with coefficients from f, that is what I mean. So, it is just written in that notation, there is no need to be, no need to panic over that notation, just writing f x is a polynomial with coefficients from f. So, one is to be little bit careful here when you try to find roots for a of x, when do you say x not is a root of a? If a of x naught is 0, x naught is said to be a root or 0 of a of x 1 of those two terms.

One very interesting result which is true is, a of x naught is 0 imply something about factoring a of x, this is true if an only if a of x can be written as x minus x naught times b of x. And then degree of b of x will clearly be one less than the degree of f x. So, this is one way to factor of polynomial so, our goal is to factor polynomial, if you want a factor polynomial one way is to is to find the 0 or root of it. Now remember, my a have coefficients from f so, where will I look for roots? In f, right in f is what you dominantly think, but remember several times when you have real polynomials, polynomials with real coefficients or let us say, polynomials with even rational coefficients, where do you look for the roots? Not just in rational, right, you look for rational, if you cannot find rational then what do you go? You go to the real field, right.

So, for instance if I have an equation of the form x square minus 2, if a of x is x square minus 2 i will belongs to q x, by the way q is the notation for the rational numbers, rational numbers, rational numbers they form a field. So, here there is no root on the, in the rationals, can you show that? Sure you can show that? There is no root on the rational, there is no square root of 2 in the rational, you can show root 2 is not rational, you can easily it can be shown. So, there is no roots in the rational so, what do you do? You go to the real numbers and in the real numbers there is a root, square root of 2 is a root in real numbers. They can be also other polynomials of the form say x square plus 2 or plus 1, which again belongs to q x, for this you would not have solutions even in the real's, you have to the complex fields.

So, it turns out when you want to find roots for a polynomial over f x, you can either look for roots in f or roots in fields that contain f as a sub field. So, if any fields contains f then you can even look at that so, we have not talked about, much about these things, we will come to it soon enough. But hopefully your intuition with the rational field and the real field and the complex field is giving you that idea, you can either look for roots in your field and if you do not find roots in your field, it is not factorable in that field. Then what can I do? I can go to a larger field, which contains my field as a sub field and then in that larger field I might find roots.

So, it turns out for polynomials for the rational coefficients, which is the largest field you have to go to, complexes you can show that interesting results, it is called the fundamental theorem of algebra. So, at any polynomial has a root in the complex field. So, any polynomial you can keep on factoring, factoring, factoring till you get all possible factors in the complex field. So, you do not have to go beyond complex field, if you have rational coefficient, in fact any polynomial with complex coefficients also has a root in the complex fields. So, it is enough, it is closed so, at that point it ends, you do not have to search beyond that.

Such things will not be true in finite fields, you will have to go on and on and on and on, eventually you can factor, but you will have to go larger and larger fields, there is not saying when you can stop it is, it is a bit of a different result in finite fields so that, something you can expect coming in. And factorization you play a fairly important role, both in the construction as well as in the, in the, in the codes that we construct. So, factoring a polynomial over field is fairly important. Now having said that the next thing is, if p is small and f is some f p and if p is very small, let us say 5 or 7 or something and if I know there is a root, factoring it is quite easy, why?

All I have to do is this, right? I have to simply substitute all the possibilities to find the 0, finding the root of a polynomial in the real line is a bit more difficult. For quadratic you can do it very easily, for cubic also there are known formulas, for quadratic there are known formulas, but from 5 onwards there is no known formula. But in the finite field that problem does not exist because you have only finite number of elements, you simply substitute one after the other, you will find out which one is the 0. So, it is a little bit simpler, but also non-trivial in some ways.

So, we will stop at this point, if you, if, if some of these things shocked you or you want to go back and read about polynomials and numbers and all that, there are several good books, I would suggest books by Rosen for number theory, Kenneth Rosen. Lots of reasonable books, there are very accessible at all levels. So, we will stop here.