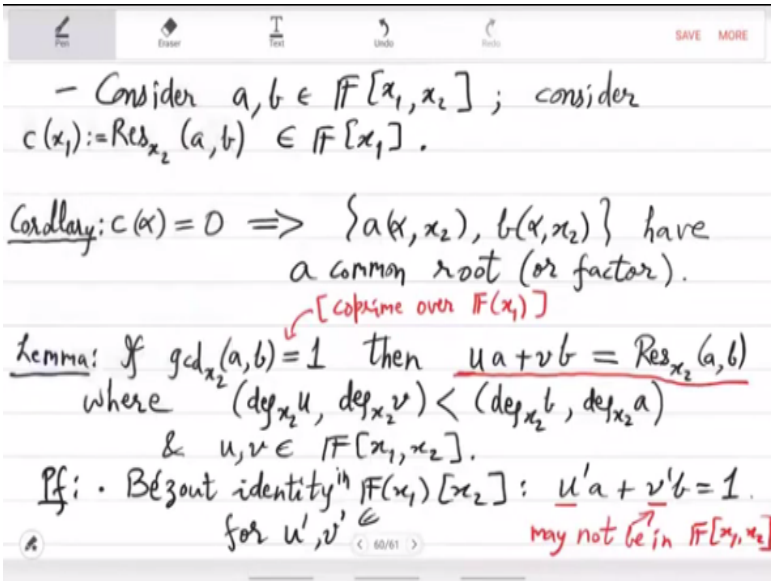
Computational Number Theory and Algebra
Prof. Nitin Saxena
Department of Computer Science and Engineering
Indian Institute of Technology, Kanpur

Lecture No -12

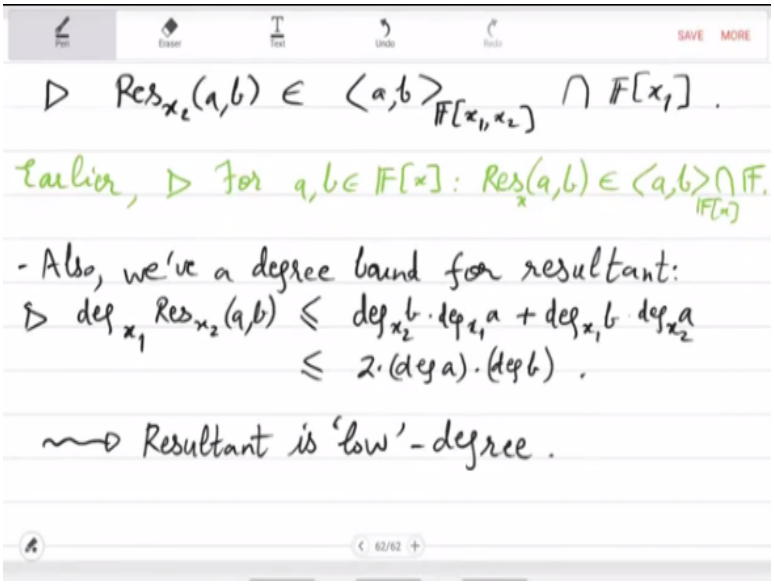
(Refer Slide Time: 00:16)



So last time we are trying to study Berlekamp algorithm as a reduction strategy. And so first thing that we did is to define resultant. For two polynomials, what is this invariant polynomial called resultant and we showed that this lemma that if g c d if a and b are co prime over the function field x1, so we are thinking of a b as a univariate in x2 over F x1 if they are co prime then this Bizout identity u a + v b will give you resultant.

In other words resultant is in the ideal of a,b and remember that resultant is a univariate in x1 to eliminate x2. So we showed that, any questions about this. So this is what we have done, so we have shown that resultant is in the ideal of a b and v also are degree limited as in the Euclid algorithm case.

(Refer Slide Time: 01:31)



So let us write it down in terms of the ideal, so resultant is in the ideal a, b over bivariate. polynomial ring and it is also univariate. So resultant is a univariate in x1 and it is in the ideal, that is the property. So when a and b are co prime over the function phase then resultant is in the ideal we have seen and what happens if a and b share of factor, they are not co prime. Why is the resultant then in this?

In that case resultant is 0, so in both the cases it is in the ideal. This is without any assumption and the earlier statement was for univariates. So for univariates resultant is in the ideal is also in the ideal a, b intersection F. Because for univariate if the univariates are co prime then the resultant is non 0, it is actually 1 can take it as 1. Why is 1 in the ideal a, b? Well because a, b are co prime.

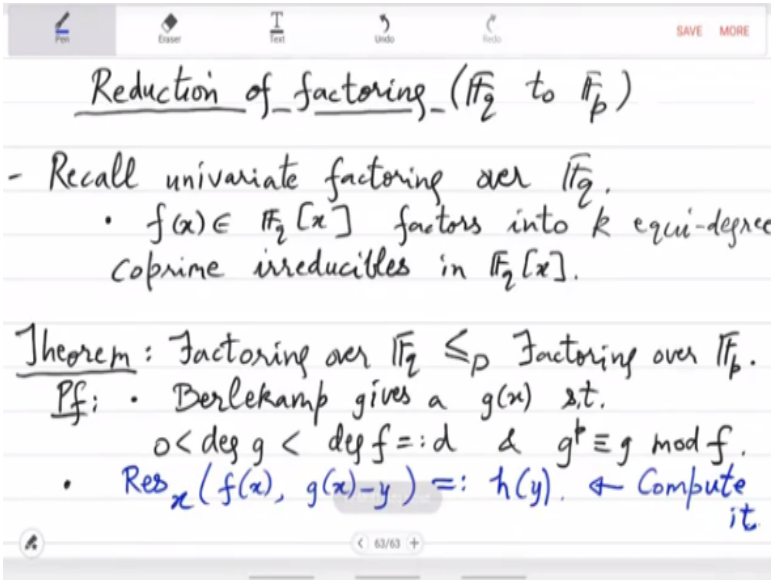
So actually we will send it everything this is happening over F x. So when a b co prime this is correct and when here we share a factor then the resultant is 0. And 0 is in the ideal, so this is the green thing is for univariate and the above fact is for bivariate. They look similar, is that clear? So this is something to remember and what is the degree of the resultant? So degree of the univariate resultant which is a univariate polynomial in x 1.

This is at most, so remember that it was the matrix given by the linear system to find u and v in Euclid algorithm. So it is related to, it is the basically the matrix dimensions are some of the

degrees of u plus degree of v. But then that, when you compute the determinant you will have to take the product, so it is kind of quadratic. So it is degree of b with respect to x 2 times, degree of a with respect to x 1 plus the symmetric thing.

So we will use the bound twice degree of a times degree of b. So if you exactly write down the matrix and check the determinant degree, you will get this bound; it is basically product of the degrees of a b, which is the total degree so degree with respect to both x 1 and x 2 all variables product of that and this factor of 2. So resultant is a we say that resultant is a low degree polynomial, It is univariate and it is contained in this bivariate ideal. So that finishes the discussion on resultant.

(Refer Slide Time: 06:55)



Next thing we will do is the reduction of factorization. So say you have an algorithm that can factorize polynomials over prime fields Fp. How can you use this algorithm to factorize polynomials in the extension plate and this is nontrivial because polynomial which was irreducible in the prime field when you go to an extension it can easily reduce. So in the prime field, you did not need to factor the polynomial but in the extension, you will actually have to find roots.

So it is highly non-trivial that these two things are actually related. On the other hand if you can factorize over F q, can you factorize or F p? Why is that? Exactly so F q is a bigger field. So if

you can find factorization over F q. Then do find factorization over fp you just have to cluster the conjugate factors. So if you for example, if you can factorize over complex, then you can factorize over real's because the conjugate of square root of - 1 is minus of square root - 1.

So you just have to cluster the conjugates and multiply them so you get a factor in the lower field. So, F p to F q is actually trivial. But F q to F p is non trivial right. So the non-trivial thing will do and then the two will become equivalent. Over the finite field fq. So you were given a polynomial f x. And this factors into. So recall all these results we had about factorization over f cube. So we can by preprocessing steps we can assume that F x factors has equi-degree irreducible.

So it factors into k equi-degree co prime irreducible in F q x. So this is the setting we are in, so now we want to find these equi-degree factors, irreducible factors given a subroutine to factor F p polynomials. This is our goal so, that is what now we will show. We show that factoring over F q reduces in deterministic polynomial time this is not prime p but polynomial time p the complexity class it reduces into factoring over F p.

Where q is a power of p q is p square p cube so on. So we will give a deterministic polynomial time reduction from this first problem to the second problem. Thus making the two equivalent. So further recall Berlekamp algorithm. So Berlekamp algorithm will give you a g such that so degree of g is less than the degree of f which we are calling d. So g is degrees between 1 and d - 1 and g raise to p is g mod f.

So this we were able to find by linear system solving. A nontrivial g says that this happens, so after, so this was step one is actually berlekamp algorithm. Step 2 is was that you will then take g c d of g - alpha with f for all alpha constants numbers 0 to p - 1. So that was the part which was an expensive, if p is large. So we will actually modify that, so, the second step which was taking g c d of f x with g of x - y, like varying y 0 to p - 1.

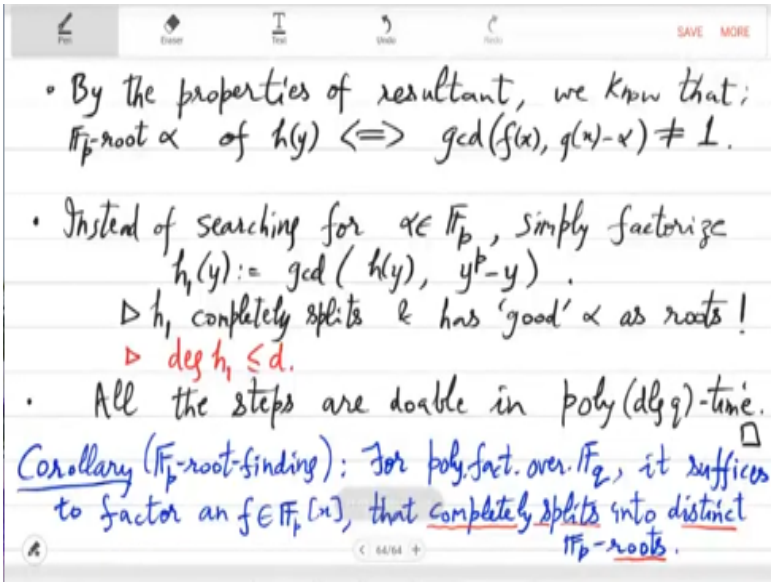
So that we will now do instead we will actually say that, look at the resultant and called this. So think of taking g c d of these two bivariantes. Why think of y the formal variable? Do not fix y to

a number? And so you are interested in g c d of these two things with respect to x. So analogously, you should take resultant of these two when you take resultant this becomes a univariate in y. So let us call this h y.

So you compute this, so whatever you have learnt about resultant using that you can compute it in deterministic polynomial time. And now what is the connection of this with factors of f. What do you know? So in the context of berlekamp algorithm, you know, that some number y would have worked and given a non-trivial g c d which means that, that number let us say alpha what can you say about each of alpha?

So those all those good alphas are actually roots of each and they are numbers right so here you can call the subroutine that factors over F p. So if you have a subroutine, if you have algorithm that factors over F p, it will factor each and it will give you the alpha. And then you happily compute step two of Berlekamp. That is the fast algorithm, so that basically is the proof.

(Refer Slide Time: 15:53)



So by the properties of resultant we know that an F p route of h always satisfies this. This F p root I am calling alpha and actually this is for any F p root, let me clear this. So any F p root alpha of h y will share a nontrivial g c d with f of g minus alpha will share and nontrivial g c d with f and any alpha for which the g c d is nontrivial has to be a root of each if and only if. Nothing is lost in each so we can make this in fact. Any questions about this?

This is the main crux of the proof that you observe g c d is non-one if and only if alpha is a root of the resultant. So you compute the resultant and factor it. This is the constructive proof. So, instead of searching for alpha simply factorize this polynomial. So you are only interested in the roots of each, not quadratic factors or cubic factors or linear factors. So you can extract them by taking g c d with this y raise to p - y contains all the roots and it is also efficiently reducible mod h.

So you first reduce it by repeated squaring mod h then take the g c d. Call this each one, so now each one only has roots. It does not have any other irreducible factor. So on this h 1 you apply your subroutine. Each one completely splits and has good alpha as roots. So one thing here is what can you say about the degree of h and h 1? So first what is the degree of h? So degree of a h is, it is not more than 2d square. In fact, it is not more than not d square.

I mean, we had some factor of 2, so it is something like 2d times d - 1. But you should also note that this situation is very special here. Why appears only in one place? So actually if you analyze it, it will come out to be d. 2 d square is that easy bound. This will actually come out to be d, so let us note that it is not very important but good to know. So it is not 2d square, but much smaller it is actually d at most d.

And then when you further take g c d it cannot increase so degree of h 1 is also at most d. So these are all low degree polynomials. So every step here, you can actually efficiently compute like in deterministic polynomial time may be even slightly cubic time you can do all this. This is all this is all happening in cubic time, sub cubic time, in fact all the steps are doable in polynomial in d log q time, so Berlekamp had a factor of p also so that is now gone.

This is really this is truly polynomial in the input size was d log q which are d many monomials in f and each coefficient log q bits. It was really d log q and you are doing everything polynomial in that, Any questions? So this is the reduction so Berlekamp is an algorithm that is practical when p is small characteristic is small. Even when it is not small if it is very large like 1000 bits or whatever.

A Berlekamp is actually a efficient reduction, it reduces the field extension factorization to prime field factorization. Then each one will be degree, each one will be 1. So nobody says that h 1 is degree one or more. h 1 just may just be 1 no factorization happens. Those things are dependent on f. So we are only giving upper bounds here. So this gives you the corollary about Fp route finding.

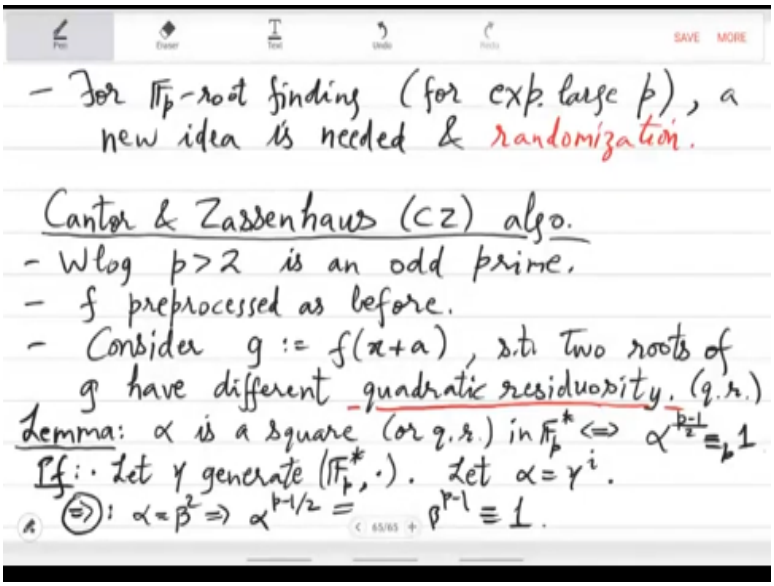
So in this business of polynomial factorization over finite fields, you now have to solve only one problem which is finding F p roots, you do not need to find quadratic irreducible factors cubic irreducible factors and so on. So you only want to find only need to find linear factors which is equivalent to finding roots and roots are remember there only numbers from 0 to p - 1. So if you can just find the F p roots, then you can solve everything.

Because of this reduction in this reduction ultimately each one only has roots and they are all distinct. So that is that corollary we note that for polynomial factoring over F q it is suffices to factor and f which is an F p x that completely splits into distinct roots. So completely splits that is an highly non-trivial thing and this thing roots. So from this point on we can forget about factor finding and we can only talk about the F p roots finding. Because any other problem can be solved using this with minimal overhead.

That is true and we always take p to be prime, any questions? So that is the state of the art of Berlekamp algorithm and all its connections. But this has still not solved our very basic problem of finding F p routes when p is very large. That is actually you will see in future applications that this is a very realistic practical problem that you have a polynomial over a prime field and you want to find the root.

For example, you may want to find square root of a number when it exists. You are given a number a mod loop p and you want to take square root of a mod loop. So either it will exist or not exist which you will be able to check but when it exists you want to find it and that you have you still have not solved that problem. So, so now we will see an algorithm that solves those problems.

(Refer Slide Time: 28:34)



So, for exponentially large P, so a new idea is needed now it is very old-fashioned and we will actually need the randomization. So this is essentially one fundamental algorithm and it is called Cantor Zassenhaus, it was given by Cantor and Zassenhaus so we will call it CZ. Yes, so we have to develop few concepts in prime field before we can describe this. So let us prove some basic properties first of prime fields.

So we will assume that p is an odd prime, why can we assume that? What happens if p is not an odd prime? Then the only possibilities p is equal to 2 and what do you do then? 0, 1; No, p is the characteristic F q can be anything above F 2. Yes, so all those problems are already solved by Berlekamp. So p = 2 any q which is a power of 2, that keep the those factorizations you already know efficient algorithm.

So those you do not have to solve and when you look at each one, then each one is only it can only have 0 or 1 value correct and then on top of that there is whole Berlekamp machinery to take care of factorization. So we will take p to be large so definitely odd and f as before. So we have already preprocessed F and we are at the point where f just has F p roots distinct and no other factors.

So you look at two distinct roots of F let us say alpha 1, alpha 2 and so we somehow want to separate them factorization is basically separating the roots. So that is separation we will do by transforming F and the way we will transform is the simplest way possible, which is by linear shift. So if alpha 1 alpha 2 are roots of F what are the roots of g? alpha 1 – a and alpha 2 minus a, so alpha 1, alpha 2 have been perturb by a.

So when you perturb these two different elements alpha 1, alpha 2, maybe their properties become different and you can distinguish such that the roots, let me be specific such that the roots of g have different properties and the specific property I am looking for is called quadratic residuosity. So, what is the meaning of quadratic residuosity? Whether a number is a square or not, right? If it is a square it is called a quadratic residue, otherwise it is called a quadratic non residue.

So alpha 1, alpha 2 are the roots of f for g they become alpha 1 - a alpha 2 - a and the question is whether the one of them is a square the other is not? So we want to make the reciduosity different by choosing any, do you know ways to check residuosity? How do you check this? Yes. Gauss's law is complicated something simpler than that we will use, we will use this thing e to the p - 1 or alpha to the p - 1 by 2.

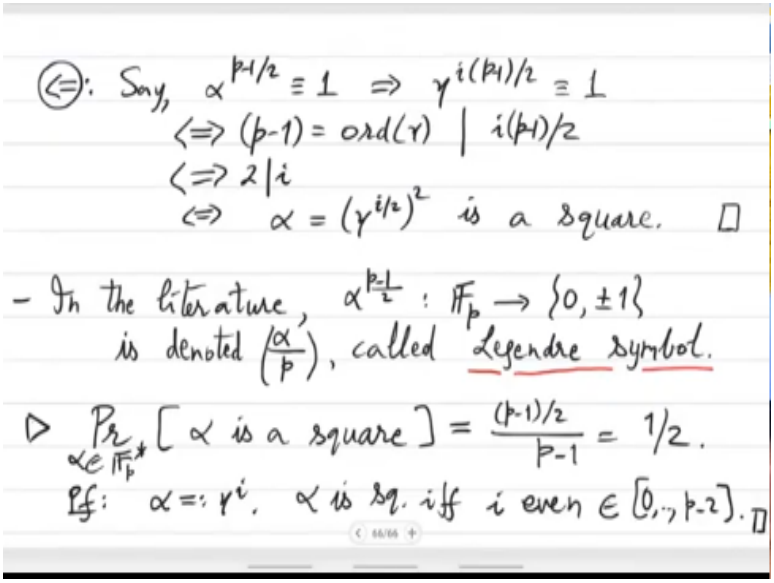
But that is the same as alpha to the p - 1 by 2. So, we show that alpha is a square or let me shorten this q, r. Alpha is a square or a q, r in F p. If and only if alpha to the p - 1 by 2 is 1 mod p, so we will show that it is very easy to test whether a number is a square or equivalently a number is a quadratic residuo. So all you have to do is raise it to p - 1 by 2, p is odd so p - 1 by 2 is an integer.

So you just multiply alpha that many times and get 1. So, what is the other value you could have gotten? Why only +- 1? Exactly, so note that by formulas little theorem alpha to the p - 1 was 1 and this is the square root of 1 and in a field there are only 2 possible square roots both of them, you know +- 1. So this value could only be + 1 or – 1, so we are actually going with + 1 and when you get - 1 then all those alphas are quadratic non residues or non squares.

So it is a very simple test, what is the proof? You can give multiple similar proofs, the easiest one is you use the cyclic nature of F p stars. So let g be a generator, g is also a polynomial. Let me not use g. So let gamma generate F p star multiplicatively and so then you can express alpha as gamma to the i. Actually one side is quite simple, you do not need gamma. So the first side if you want to show forward implication.

So suppose alpha is a square beta square then, what can you say about alpha to the p - 1 by 2? So that is beta to the p – 1 which is 1 of the forward implication is clear. RHS has to be 1, let us now look at the converse we will need gamma there.

(Refer Slide Time: 38:28)



So see alpha to the p - 1 by 2 is 1 which means that gamma to the I is 1 which means, so what is the order of gamma? So p - 1 has to divide it, because I mean, no number below p - 1. Essentially, you have since the multiplicative order of gamma is p - 1 gamma raised to anything if gamma is 2 M is 1 then M has to be divisible by p - 1 by the order. So, let me write it that way so, p - 1 is the order of gamma and that has to divide i p - 1 by 2.

This is in fact if and only if which implies that i is even, this is also if and only if; So, which means that alpha is a square, so this in fact is all you needed this is a proof for both sides. So kind of this discrete log of alpha with respect to gamma is I, this i has to be even the i is even if and only if alpha is to p - 1 by 2 is 1 which is then equivalent to saying that alpha is a square.

Any questions? So in the literature the number theory, this plays a very important role alpha to the p - 1 by 2.

This is a very important function of alpha, so think of the finite; the prime field being fixed to a fixed by p and then this is just a function in alpha there from F p 2, F p, in this is from F p 2 what is the image? This is from F p to 0 +- 1 takes the only 3 values and it has these nice properties that it is multiplicative. So this function for alpha this function for the beta and then this function for alpha beta those three things are related by multiplication.

So this is a very important so this is a character of F p and it is called it is denoted also by this notation alpha divided by alpha by p and bracket, it is denoted by this symbol and it is called the Legendre symbol. So this is the Legendre symbol, it is a character function of this group F e star multiplicative. So sometimes we may use this notation alpha by p is a short hand when p is not a prime number let say p is a general composite then also this symbol is defined it has a different name so that we will use in the future. So for now just this is just a basic thing we are introduced.

So as you vary alpha how many times is this symbol + 1 p - 1 by 2, so around half the time this is + 1 around half it is - 1 and it is a famous conjecture or in a way also proved that this is a random phenomenon. So when you look at very very large p so as you are going from alpha equal 1 to alpha equal p - 1 the way this + 1, -1 distribution is coming this distribution is nearly random and this is proved by proving the Riemann hypothesis over finite field.

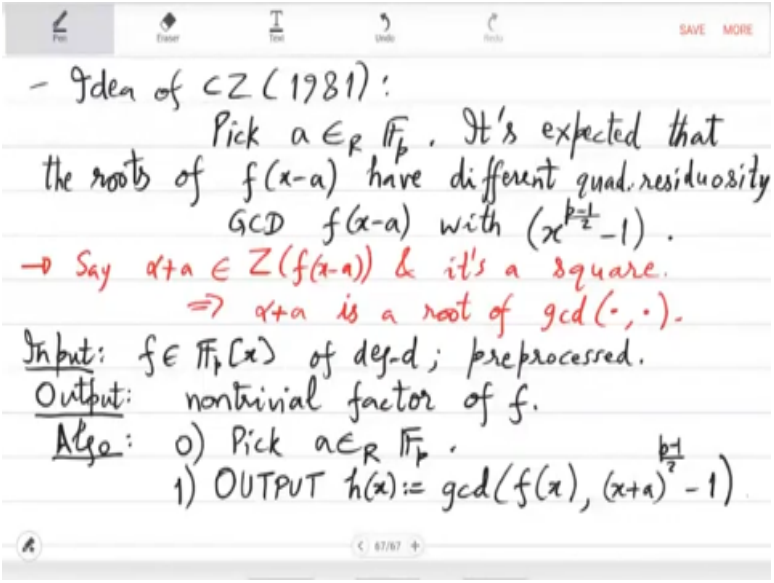
So this randomness statement is actually Riemann hypothesis statement there is a proof for that but here we only need this probability estimate probability over alpha is that alpha is a square. Let me make it a F p star will never take alpha 0. Probability that this is a square is what? p - 1 by 2 divided by p - 1. Maybe let us go back to F p, p which is less than half. So, would that is not correct.

Then I have to take p - 1 by 2 + 1. So over F p star it is half are square and half are non squares, this is so we note this proof of this is simply you will write alpha as gamma to the i gamma is the generator before as before and note that alpha is square if and only if i is even for half the time 0

to p - 1. Here actually 0 to p – 2, so this i actually runs kind of mode p - 1. So here it is going from 0 to p – 2, correct so this p - 2 is odd.

So in this interval, you have half evens and half odds. So, that is the distribution here because gamma is multiplicative order is p - 1. So here the arithmetic happens mode p - 1, this is important. This is this mode p in the base changes to mode p - 1 in the exponent. So with this under with these properties now we can actually see Cantors Zassenhaus easily. Cantors Zassenhaus is directly based on these ideas.

(Refer Slide Time: 47:25)



So the idea is that you pick a random element a, this notation means that you are picking a from 0 to p - 1 by flipping coins. So how many coins will you have to flip to discover a? So a has by in fact elements in F p require log p bits, each bit will cost a single tossing of a coin. So a will require log p random bits which is not too much if you have a coin you can flip it those many times.

So then you will be you would have picked a, so then it is expected that the roots of I am making it f of x - a now, the tool is let me change before. So the roots of f of x – a have different residuosity. So see original roots for alpha 1 alpha 2 new roots are alpha 1 + a and alpha 2 + a and say these two roots have opposite residuosity. One is a square the other is a non square. you know how to test for square, how do you test that?

So you will take g c d with x to the p - 1 by 2 – 1, so because of this property that you saw before alpha to the p - 1 by 2 – 1, you know that any square has to be a root of this and this has exactly p - 1 by 2 roots, so the roots of these actually are clustering all the squares. So if you take the g c d of this with f of x - a then alpha 1 + a will come out if and only if it is a square. So if you can distinguish alpha 1 + a from alpha 2 + a by which residuosity right then they will be separated, so f will be factored again.

I mean f of x - a will be factored, but since you know a, f is also factored is that clear. Here repeating will not help probably and like that. So, let me write it say alpha 1 + a alpha 2 + a or you can only look at 1 root actually 2 or not important. So say alpha + a is a root is a 0 of this f x - a and say it is a square. So, do you see that alpha + a will be a root of the g c d also? So the square the roots of the 0's of f of x - a which are squares we will all be clustered out when you take this into g c d.

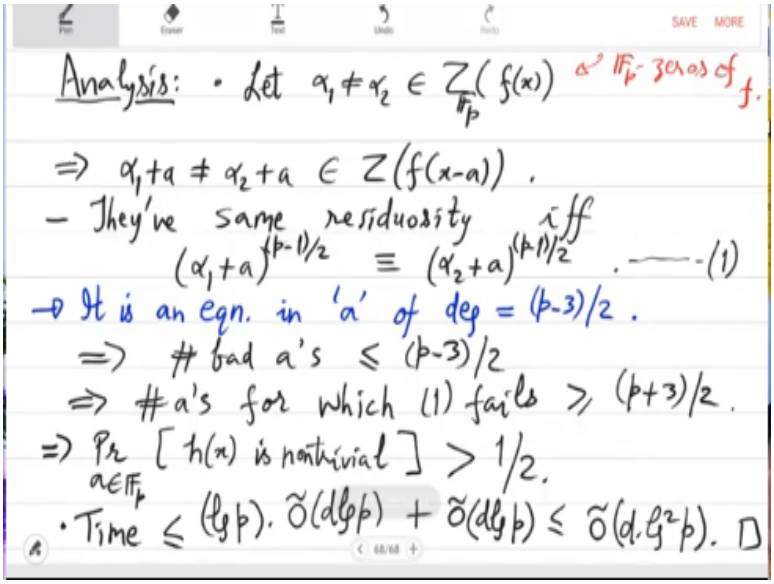
So if in f of x minus say there are two kinds of roots or 0's, then you factor it. So the your goal is to pick an a such that f of x - a has a different kinds of roots. So at least 1 root should be there, which is a square and at least 1 root should be there, which is a non square. If all of them are square, I mean this game you could have played already with f x. So in f x, if so if there is a root, which is a square and there is a root which is a non square both of them are there then when you take g c d with the x 3 p - 1 by 2 -1, f would factorize.

So actually the bad cases when f has either only squares as roots or only non-square as roots. In which case you will shift by a and try to make them different? If they are different than g c d factors, so that is the algorithm. So Cantors Zassenhaus algorithm is very simple to implement. So you are given in the input a polynomial in F p x of degree d and it is preprocessed. So all the usual assumptions, hold in the output you want to give a factor and it is essentially has only 1 step.

So this 1 step is output, the g c d of f of x - a with x to the p - 1 by 2 – 1, it is just a factor of f of x - a let me slightly change this. So, I mean, if you can take g c d of f of x - a with that

polynomial then you can as well take g c d of f with x + a to the p - 1 by 2. So that is what we are taking and whatever answer you get you output it. So if your choice of a was lucky then in f of x - a, there will be 2 roots which have opposite residuosity and with that luck g c d will factorize f. So there is only 1 step. So, let us analyze this.

(Refer Slide Time: 56:22)



So in the analysis, I will prove both the things because it is not clear whether such any even exists. That if it does not exist then random choice will have no meaning I mean, it will not help. Second issue is about efficiency so suppose a exists, but only 1 or 2 a exist. So then by random choices, you will never find them it is impossible to find them. So at this point both the existence and the densities unclear. So we will prove both these things actually in one step in the analysis.

So, let alpha 1 different from alpha 2 be the 0's of f x obviously everything in F p. So this is the notation I use for 0's and so then this means that alpha 1 + a different from alpha 2 + a both of them are 0s of x - a and suppose they have the same residuosity so, what is the equation for that? Alpha 1 + a to the p - 1 by 2 is the same. So you when you multiply it p -1 by 2, two times you get the same sign +-1.

Think of this as an equation in a, unknown a. What is the degree of the equation? -1 because the highest power of a will also cancel out, so view this as it is an equation in unknown a of degree p - 3 by 2. Now for an equation of degree p - 3 by 2, how many roots can be there, which is very

small? So this means that number of bad a's is less than equal to p - 3 by 2, which means that number of a's that discriminate alpha 1 from alpha 2.

Let us precise number of a's for which one fails is a lot and so that is we can you can also consider equal to 0 will not change anything. So number of bad a's above is limited by the degree, so number of good is which means that they are able to discriminate alpha 1 + a from alpha 2 + a by a residuosity, that is p + 3 by 2, which is at least half. So more than half of the a's are good.

So not only do they exist they are densities also 50%, now it does not matter actually where they lie in this interval 0 to p - 1, it does not matter where they lie if you randomly pick with more than 50% chance it is a good. Why the degrees so, what is the degree of a in the LHS? p minus 1 by 2 and the a raise to p - 1 by 2 term both sides cancel so it is 1 less. it is not an important point in time it is even with p - 1 by 2 degree you get essentially the same thing that density of good a's is half.

So what you deduce is that probability over a in F p that the h x that you are outputting, h x is nontrivial given that F factors. I mean, that is already the pre-processing so obviously a F factors so let me not even say that, this is more than half. So probability that for a random a Cantors Zassenhaus algorithm will factor F. this is at least half and time complexities clearly polynomial in log p.

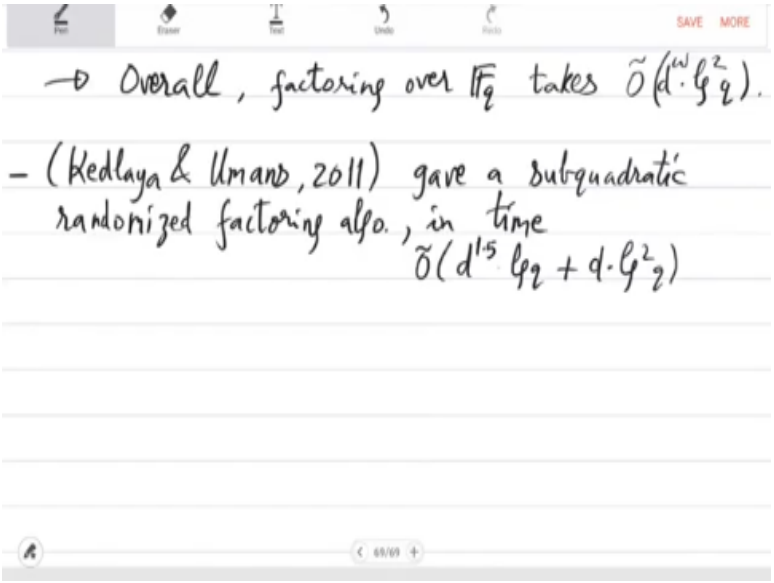
In fact you can even calculate it other than the pre-processing step this is just g c d so its again quadratic time or slightly move in log in d log p, so this is a very fast very practical algorithm in fact one of the only algorithms useful in practice to factor any polynomials over any finite field. Here, we can write the time also, so this is for repeated squaring I mean again this g c d computation is possible only because of repeated squaring, this is a very special polynomial explicit of the p - 1 by 2, it is exponential degree.

But then it is a single exponential so you can compute it by repeated squaring mod f So you square then you divide by f then you again square and again divide by f. So that will go on for

log p steps, so there are log p multiplications and then division and then g c d. So log p times linear in d log p plus the eventual g c d is O tilde d log p. So which is O tilde d log square p. So it is nearly quadratic in log p in d it is actually linear time.

So it is a very fast algorithm, although there were these pre-processing steps which were expensive. Pre-processing took around cubic, that is for the g c d. The d log p after the plus for g c d. The first one is for reducing exponential degree, second polynomial. This x + a to the p – 1 by 2, that is the dominant complexity.

(Refer Slide Time: 01:05:35)



Yes, so if you look at the overall factoring algorithm takes around there is d raise to Omega in the pre-processing it was actually d cube, but I was mentioned that if you use the fast matrix multiplication methods, you can actually make it faster. This will be something like d raise to 2.4 or 2.41 and there is there are obviously log q terms. I forget maybe you will do in square so this much time, it is sub cubic.

If you want to so when you combine all these components of acquisition this irreducibility testing, then Berlekamp and then Cantors Zassenhaus overall there are improvements to this. So Kedlaya and Umans; few years ago gave a sub quadratic time method. So it is time complexity is something like d to the 1.5 log q + d log square q. So this is instead of d to the 2.41 this is d to the 1.5, it is sub quadratic it is still not linear time, but this is the best known.

So polynomial factoring from the very beginning finite fields from scratch you can solve in this first time, any questions? Probably when you randomness is only from the speaking of random points in F p, so that is in too much it is the other linear algebra steps etcetera, which is costly.