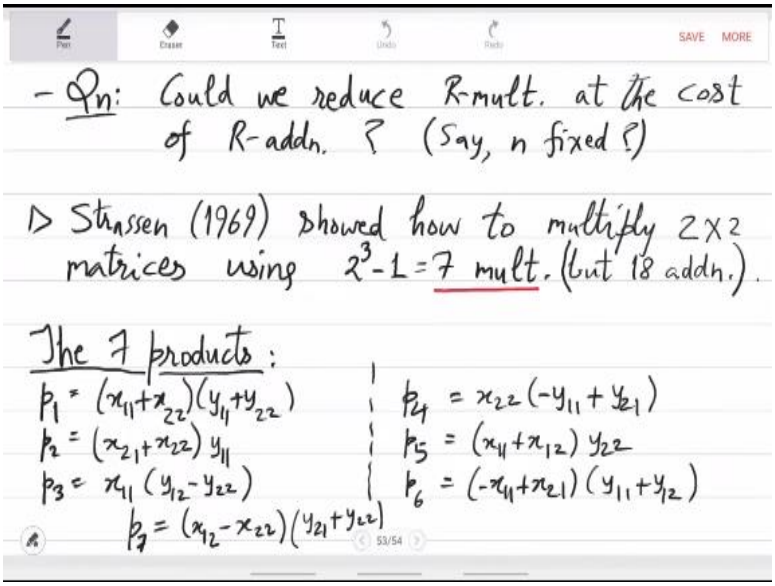
**Computational Number Theory and Algebra**
**Prof. Nitin Saxena**
**Department of Computer Science and Engineering**
**Indian Institute of Technology-Kanpur**

**Lecture – 08**
**Matrix Multiplication Tensor**

Okay, so last time we introduced the problem of matrix multiplication.

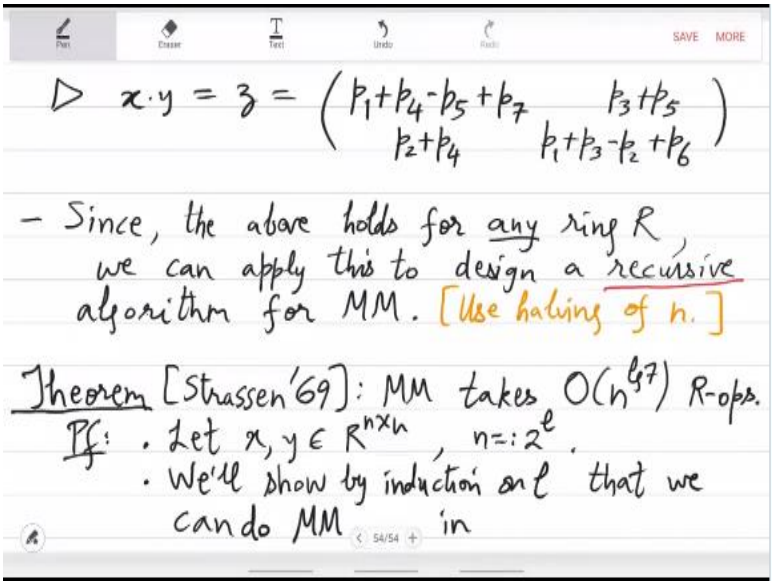
**(Refer Slide Time: 00:18)**




And in fact, we also gave these 7 products of Strassen. So p 1, p 2 to p 7. So these are products not of just two variables, but actually 2 linear forms. Left is an x matrix and right is in y matrix. So there are 4 x's and for y's. And when you compute these products, you have to take care of the order. Because remember that x 11 may itself be a matrix and y 1 1 may itself be a matrix.

So that product is actually non commutative. So p 1 is exactly that. It is not the reverse. You cannot do y 11 plus y 22 times x 11 plus x 22. It is exactly in this order.

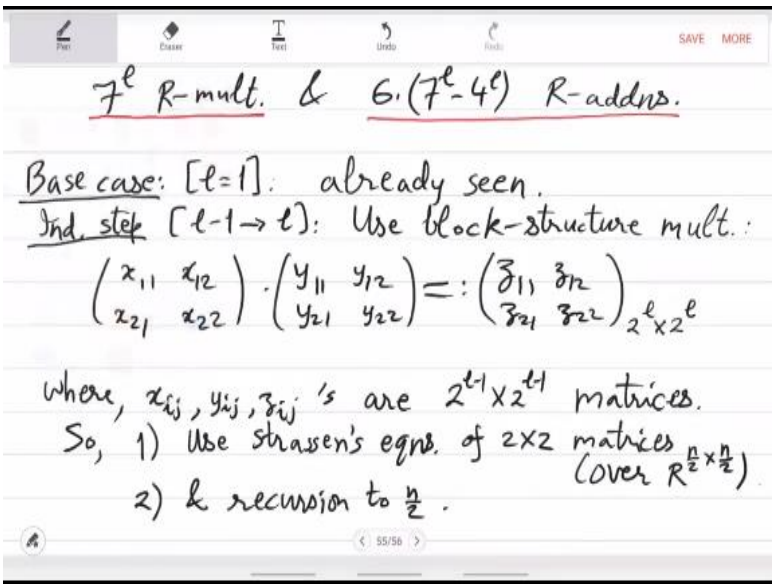
**(Refer Slide Time: 01:11)**




And with that, you can actually check that x dot y will be this without using commutativity of x ij and y ij. So today we will finish this formally the proof of Strassen that matrix multiplication takes n to the log 7 operations where the ring R is the is where the entries of the matrix come from of the n cross n matrix. Any questions? Okay, so x and y are n cross n matrices over R. We are assuming n to be a power of 2.

So proof will be an induction on l, okay. So we will compute how many multiplications and how many additions in the base ring R are required. So we will show by induction on l that we can do matrix multiplication in

**(Refer Slide Time: 02:51)**



So 7 raised to l R multiplications and that many R additions. So actually both multiplications and additions are not too many. This is n to the log 7. That is the order. So base case is l equal to 1, right. So these are 2 by 2 matrices which we have already done. So for l equal to 1 you got 7 multiplications and 18 additions, right. That you have that is the base case already.

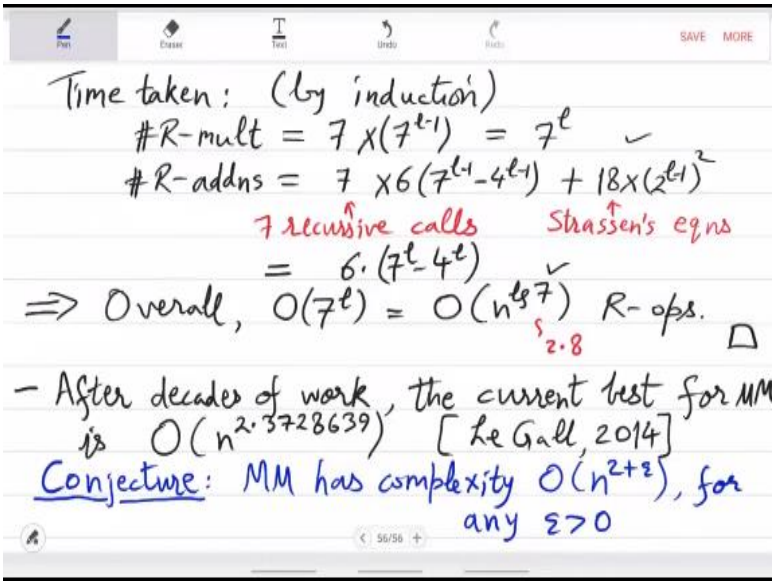
That was actually the starting point of all this. The induction step is l – 1 to l where l is at least 2. So how do you go from, how do you take this one step? So this will be the block structure application. So you look at x 11, x 12, where these x ij's, y ij's, z ij's they are 2 raised to l – 1 cross 2 raised to l – 1. And the full matrix is 2 raised to l cross 2 raised to l, right. So for n cross n matrix, you are looking at the block structure n by 2 cross n by 2.

And so this n by 2 multiplication of matrices, multiplication of n by 2 by n by 2 matrices this you do inductively or in the algorithm recursively. These are 2 raised to l - 1 cross 2 raised to l - 1 matrices. So there are two things. One is you when you have to multiply x ij with the or x ik with y kj that multiplication you will do recursively and assuming that yeah, so that is the recursive part.

And this also will require then this 2 cross 2 matrix multiplication. For that you use Strassen equation. So well I think look at this look into two steps. So first is use Strassen equation here. And the entries of these matrices are in which ring? So that is n by 2 cross n by 2. So n by 2 cross n by 2 matrices over the ring are obviously also form a ring. So you do this 2 cross 2 matrix multiplication over this ring using Strassen's equations.

But that in the end will require you to multiply 2 such matrices, right like this x 11 plus x 22 times y 11 plus y 22 was p 1. So that multiplication you will do recursively. So do that in those two steps. And now so the algorithm is it clear? Right, so it is a recursive algorithm using Strassen's equation. So now you solve you analyze the time complexity.

**(Refer Slide Time: 08:02)**




So that will finish your induction step. So what is the number of R multiplications? So R multiplications that you are doing? So that is 7 to the l – 1 many multiplications, right. That is for n by 2 times n by 2 matrices and how many such multiplications you are doing in the induction step that is given by Strassen's number which is 7. So that 7 to the l, which is the correct value in the to go to l.

And look at number of R additions. So that number was 6 7 l – 1 4 l – 1 by the induction step. And how many such additions are needed? So let me write the expression first. So I am giving it in two parts. First look at 18 times 2 raised to l – 1 square, right. What is this? So these are the 18 additions which you do in Strassen itself, right? That is the number 18. And all those are n by 2 by n by 2 matrices.

So that many ring R additions. What is the first part? The first part is for the recursive calls. You are making 7 recursive calls and in each recursive call that many additions. And this is in Strassen's equation, okay. So that is the overall additions you need in the base ring R, which you can see is as required in the induction step, okay. So the second part is 4 to the l – 1 times 18. So that you will subtract 42 from that.

The first term is main term is 6 times 7 to the l. Is that clear? So overall, we have shown 7 to the l ring operations, which is n to the log 7 to base 2. Okay, that is Strassen's matrix multiplication theorem. But you have precise expressions for multiplications and additions in the very base ring. Any questions? So yeah, one could

already be happy with this. This will give you some advantage if you take bigger and bigger matrices, bigger and bigger n.

So it is a surprising algorithm improves over n cube. So this is now n to the 2.8 right, but the question is, why did this algorithm exist? Right, so by definition, it should have been n cube. Why did it reduce to n to the 2.8? So that study is still ongoing. So after decades of work, the current best for matrix multiplication is n to the 2.3728639. Okay, I give you 7 digits, because each of these digits people have worked very hard to improve.

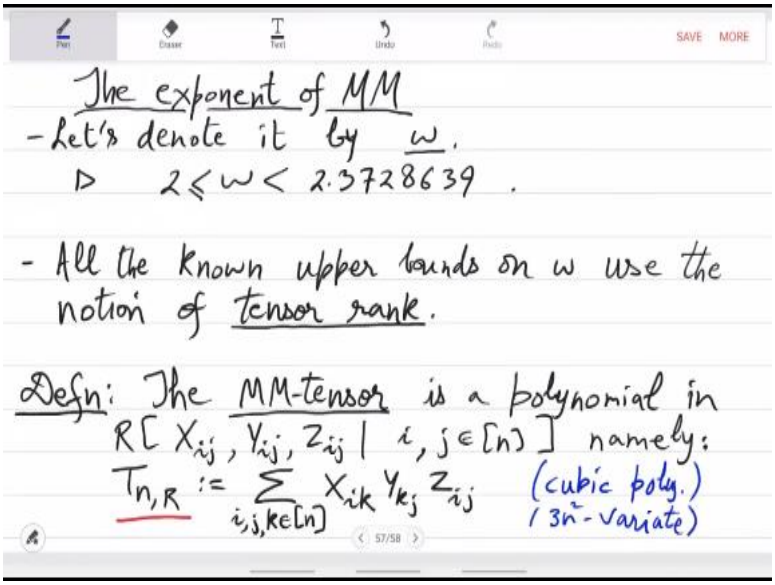
So there are at least seven papers and there are at most 50 papers on matrix multiplication. They all trying to improve each and every digit here. Okay, so slowly three has been reduced to something like 2.37. So this is from 2014 from Le Gall. So that survey paper if you want you can read and currently there are no good techniques to improve this. Whatever techniques have brought it down to this are more or less stuck.

Nevertheless, the conjecture is that sorry, exactly. So the conjecture since the time of Strassen has been that matrix multiplication has complexity n to the 2 plus epsilon for whatever epsilon you want for any positive epsilon, right. But this is still there is a gap in the current understanding. So there is still a gap of 0.37. Note that you cannot do better than this because your input size is 2n square, right.

So you can never improve on n square and your output is also n square. So you can never improve on n square. So that is really asking for too much. Yeah, after this log 7, things become quite complicated. So that will take several lectures if you want to improve over log 7 to anything. Okay, so instead of attempting that, we can just see the concepts, which go into improving log 7.

Because that concept is also helpful in other places. It is a fundamental concept in algebra.

**(Refer Slide Time: 15:36)**




So you want to understand the exponent of matrix multiplication. So let us denote the exponent by omega, not to be confused with omega, which we were using before, the root of unity. So this omega is just a number expected to be something somewhere between 0 to 2.37. So it is known that omega is between 2 and 2.3728639. Actually, we are talking about the limit. So we can also call it 2.

This 2 just means that you can go arbitrary close to 2 but never reach it. So that limit is what we are calling omega. So all the known upper bounds on omega, use something called tensor rank. Okay, that is the algebra concept, which you want to now introduce. So have you heard about tensor rank? It is also used in artificial intelligence and ML applications with tensor decomposition, what is that?

What is the simplest example of tensor rank? Yeah, matrix rank is the simplest example of tensor rank. Matrix rank is just order 2 tensor rank. So matrix is actually an order 2 tensor. And then you can keep on growing the order. So when you go to order 3 tensor it is a matrix in 3D, very explicitly. That explicit representation actually has no use, but if you want to just look at the extension of a 2 dimensional matrix, then it is a 3 dimensional matrix.

So our notion of rank for that is complicated. So for matrix rank is kind of a unique definition which is very useful everywhere. But for order 3 tensors that is not true. So there, there are multiple definitions and neither of them are very helpful. I mean,

questions they are all open questions. So order 3 and beyond it becomes complex. So the, so this omega is also related to some tensor rank.

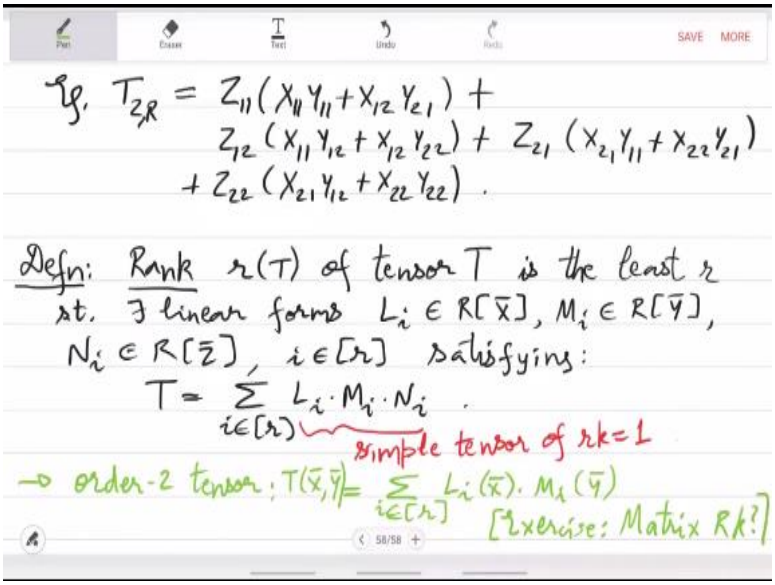
So let us first define an order 3 tensor whose rank omega is. So let us directly go to the definition. So let us define a tensor. In fact, tensor motivated by matrix multiplication. So that is called the matrix multiplication tensor. Okay, so this tensor you can think of as a 3D matrix, if you want to see it explicitly. So but I want to see it as a polynomial. So MM tensor is a polynomial in many variables.

So I will call them big X ij, Y ij, Z ij, ij 1 to n. So the polynomial itself is looks like this. So we will call it T n, R. So if you have seen the definition of this, what is that? The tensor is the tensor polynomial for matrix multiplication is, so it is directly inspired from the definition of matrix multiplication. So it will look like X ik, Y kj, Z ij okay. So definition of I mean this the tensor lives in a completely different place from matrices.

It is the roughly there are roughly n square many variables X ij, Y ij, Z ij. And in this roughly 3n square many variables you have this cubic polynomial okay, this cubic polynomial T n, R. In this if you look at the coefficient of Z ij then you will see an expression which is similar to the expression for small z ij that you saw in matrix multiplication okay. So that is the connection.

Or that is the relationship between this definition and matrix multiplication definition. But, we are doing very different things here. So we have defined this cubic polynomial. So its degree is 3. And it has many variables.

**(Refer Slide Time: 22:17)**




So for example, if you look at the matrix multiplication tensor for 2 by 2 matrices, so it is Z 11 plus then Z 21 plus the last is Z 22. Okay, so there are exactly three times 2 square many variables. Yes. Is it? We do not want it to be ordered. **"Professor - student conversation starts"** And also we will need an extra condition for k to be in between i and j. **"Professor - student conversation ends".**

No, that is not the case in matrix multiplication. K varies from 1 to n. Yeah cubic polynomial and 3n square variate. That is true. Any questions? And so then for 2 by 2 matrices you can see that there are 12 variables three times 2 square. And you have these cubic monomials in X, Y, Z. And you have 8 of these monomials. **"Professor - student conversation starts"** Sir, what is the connection of these two matrix multiplication? **"Professor - student conversation ends".**

Yeah, I know you want to hurry. We are going there. So once this definition is there, let us look at the rank of this tensor. So rank of a tensor in general and this tensor in particular, the most common rank defined is you want to decompose this tensor into sum of simple tensors. By a simple tensor, we would mean one where you are just multiplying linear forms.

So linear form in X times a linear form in Y times a linear form in Z, okay. For example, X 11 times, Y 11 times, Z 11. This is a simple tensor. This tensor is called rank 1. So you want to decompose your tensor into a sum of simple tensors. The minimum number of these simple tensors is called the rank, okay? So I will give you

the formal definition afterwards. But with that in mind, what is an upper bound on the rank of tensor T to r? How big can that rank be?

Can it be 9? Yeah already, just by brute force, you can see by definition, T to r is a sum of 8 monomials. Right, so its rank cannot exceed 8, 8 is the maximum. But you are not sure whether it is actually 8. It can also be 7. And it was Strassen's imagination that he actually gave a decomposition where it is 7, okay. And there is a proof that it is not 6. So the rank of this tensor is exactly 7, okay. That is the reason for log 7.

So rank of Tensor T is the least number r such that there exist linear forms. So L i which is purely in terms of x variables. Then M i which are purely in terms of y variables. And N i purely in terms of z variables. So these are linear forms; i will range from 1 to r. And all this data has to satisfy the following equation. The tensor is decomposes as, it decomposes as L i times M i times N i, okay.

So rank of a tensor, so this now you can see that it is a pretty general definition. It does not have anything to do with matrix multiplication tensor. There is no need to restrict to matrix multiplication tensor. You can look at any polynomial in clearly demarcated variables. So family x, family y, family z. And you can ask the question whether that polynomial your polynomial can be written as sigma L i, M i, N i.

So this is the rank of an order 3 tensor. And if you have order 4 then you will have 4 variables in the partition. So you will have let us say v bar, x bar, y bar, z bar. And then you have to decompose it into product of 4 linear forms, okay. So maybe K i, L i, M i, N i. So this is the general concept of tensors, tensor decomposition and rank. So this is a simple or rank 1 tensor.

So you want to decompose your tensor into few simple tensors. If you cannot decompose into few simple tensors then your tensor is hard. Okay, that is the understanding. But it is also true that in life it is hard. It is hard to find hard tensors, okay. So this is not an easy problem to come up with a polynomial and show that its tensor rank is high. So these are classic lower bound questions and mostly they are open questions.

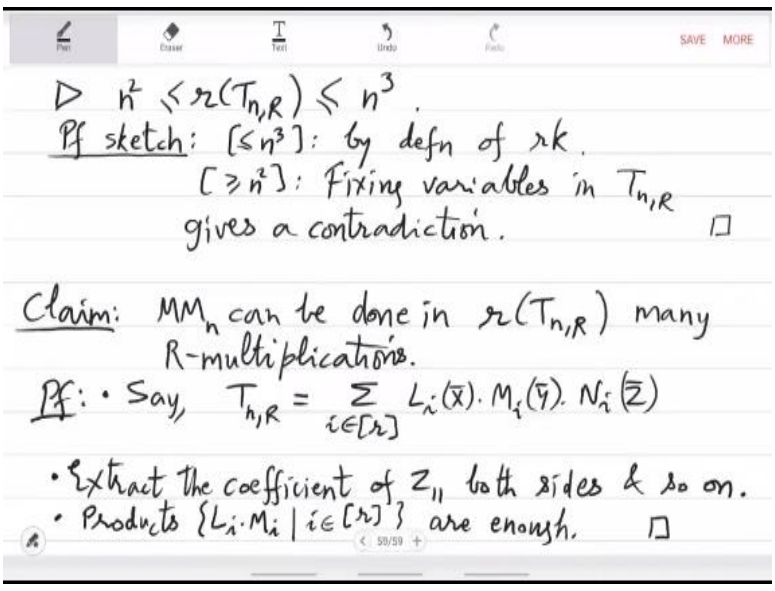
Yeah, actually examples are not there. An explicit polynomial with high tensor rank that is an open question. No, before we jump to permanent this class may not know it, something even simpler is so what is the meaning of order 2 tensor rank, right? That is an instantiation of this or specialization of this. So order 2 tensor let us say you have some tensor T in two families of variables x bar, y bar.

And you want to write it as sigma L i times M i. So this is a polynomial in x bar y bar. So for a polynomial T x bar, y bar and the tensor rank to be defined this way, smallest r, can you see that this is just matrix rank? What is the matrix here? For this order 2 tensor T how will you define the matrix? Or for a matrix how will you define T equivalently?

So you are given a matrix, how do you see it as a tensor, order 2 tensor polynomial? So this, so I claim that these two theories are the same at order 2. Tensor, tensor decomposition, tensor rank is the same as matrix, matrix rank, okay. But what is matrix decomposition that we are talking about? So that equivalence I leave as an exercise. So what is the relationship with matrix rank?

So it would be a specialization of this much more general theory. But we are not there. So we are actually now at order 3 tensors. And we are looking exactly at matrix multiplication tensor. So let us move forward with that.

**(Refer Slide Time: 33:32)**



So the rank of matrix multiplication tensor for n cross n matrices over r ring R this is at most how much, n cube right? That is by definition. And it is at least how much? Yeah the at least part we have not given a proof yet, but you can show that it is n square. So how we actually how will you show that? So less than equal to n cube is simply by definition of rank.

Greater than equal to n square you can show by so you can express, if you can express T as a sum of only n square many, or let us say n square minus 1 many L i, M i, N i. Then show that you can essentially make all these products zero by fixing the variables while the tensor will not be zero. The reason for that is that in your the variables each of the variables x, y, z family have has n square many variables.

So intuitively if you set n square minus 1 variables to zero still something remains. Not the whole tensor vanishes. So it is by fixing variables. Fixing variables in T n, R gives a contradiction. Yeah, I would not go into the detail of this. You can do it as an exercise. It is not very important. More important is the connection between rank and matrix multiplication.

Let me make the following claim that matrix multiplication for n by n matrices can be done in the rank of this tensor many multiplications. And that is a very useful thing to know because, if you decompose your tensor into few products, simple tensors then you also get an algorithm for matrix multiplication. Specifically for n cross n matrix multiplication, you get a faster algorithm.

And then using the block structure you can generalize it. In fact, that is what you did for n equal to 2. So what is the proof of this? z equal to 1? Yes, so you look at the definition of the tensor and these L i, M i, N i. So say you have a representation of T into few products. So in this in particular, look at N i z bar. So these are the variables z 11 dot dot z n n. So extract the coefficient of z 11 from here.

So if you extract the coefficient of z 11, you will see that on the LHS you would have that matrix multiplication entry of Z 11, matrix of the matrix Z you will have Z 11 entry. On the RHS, what will you have? You will have a sum of few L i, M i products, right. So you have kind of Strassen's equations that you multiply r many L i,

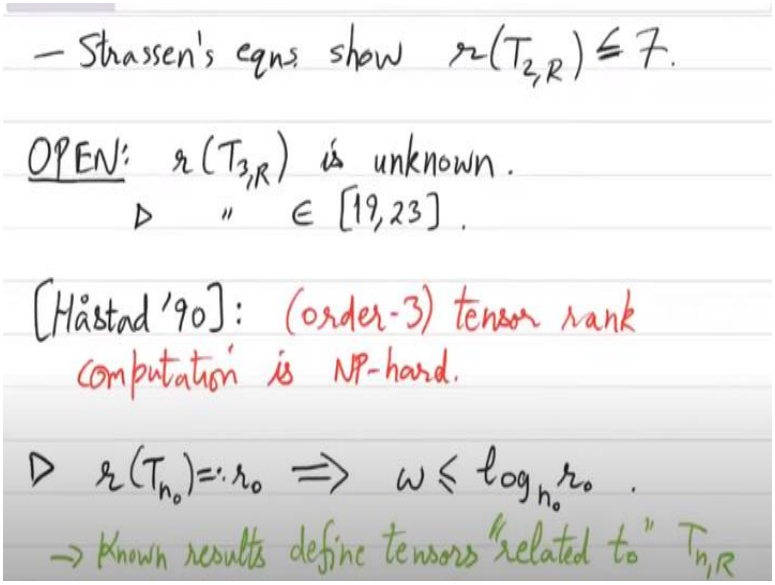
M i. And from those products a linear combination gives you the first entry of matrix multiplication, matrix product.

Now you try to extract Z 12. So compare the coefficient of Z 12 on both sides of this equation. And again you will see that same products r products their linear combination gives you the second entry of matrix product and so on. So it just follows from this equation, it is already tailor made. The products L i times M i, r of these are enough, okay. So for 2 cross 2 matrix, this is what Strassen did, essentially that he first wrote T as sum of 7 products instead of 8.

So that was a, that was saving one product. And then by extracting these coefficients, Z 11, Z 12; Z 21, Z 22 you get all these 4 expressions in terms of 7 products. And the same thing in general will work. You will only have to do r multiplications. Little r multiplications in the base ring. Is that clear? Okay, and this proof? So what is the converse of this?

That if you can do matrix multiplication in let us say, r prime many products, what can you say about the tensor rank of the MM tensor? Is the same proof. You get, then a decomposition with r prime simple tensors. Right? So this equation actually is a tight connection between how many multiplications. How many multiplications are required for matrix multiplication and tensor rank?

**(Refer Slide Time: 41:47)**



Okay, so the tensor rank of T 2 is 7. Okay, this is another way to read the Strassen's complicate equations.It is just showing that instead of 8, it is 7. So it is conceivable that you compute tensor rank of now T 3, R and then T 4 and T 5 and so on. And at some point you get something better than log 7, right? So that would be a simple minded way to improve matrix multiplication.

Yeah, but you can show that it is greater than 6. That is a separate proof. So we can maybe write less than equal to here. But it is known that it is 7. It is not less than this. Yeah. So when we say multiplications yeah so technical term for that is we want to count the bilinear complexity. So bilinear complexity means that given this matrix X and matrix Y we are only allowed to multiply linear forms.

So nothing fancier than that is you are allowed to do. That is actually a reasonable assumption because ultimately, you have to compute Z matrix whose expressions are only bilinear expressions in x and y. So maybe you are confusing or you are thinking about circuit complexity. But this is not that advanced. This is only looking at products which are quadratic in x and y. Yeah, so you can try to do that.

But believe it or not, this becomes very hard even for T 3. So compute the to compute tensor rank of T 3, the problem is that you have this tensor polynomial is how many variables. 27, right. So it is 27 variate cubic polynomial and so when you will try to discover L i, M i, N i these are all linear functions in 27 variables. So there is a huge number of unknowns for the coefficients of L i, M i, N i.

And if you try to do this by hand which means you program and you use supercomputers and all that, you still cannot make it work. Even with the current computation, it is a very hard problem to actually find L i, M i, N i and hence R. So this is still an open question. Even for T 3 this is unknown. So obviously, you know that rank of T 3 is at most 27. But we do not know its exact value.

After lot of work, what is known is that this is between 19 and 23, okay. So it is at least 19 and it is at most 23, which is an improvement over 27 but still 19, 20, 21, 22 to 23 among these five values there is still a confusion, okay. With all the computational power this is open. So that is in practice. Theoretically there is a good

explanation for this. So Hastad showed that tensor rank computation for even for order 3 tensor.

So tensor rank decomposition or actually I should say computation is NP hard. So if you are looking at a general order 3 tensor which means it has n variables. Then as n grows this is an NP hard problem, okay. So in general you can never solve it in a reasonable amount of time. What can you say about order 2 tensor rank computation? How hard is that? So that we claimed is just a matrix rank, right.

And matrix rank you can compute very efficiently. So order 2 tensor rank computation is very simple, but order 3 when you go to order 3 it becomes NP hard immediately, okay. And forget about higher orders. That is just harder. Those are the important points. What is the exact connection between matrix multiplication exponent and the rank? So that you can deduce.

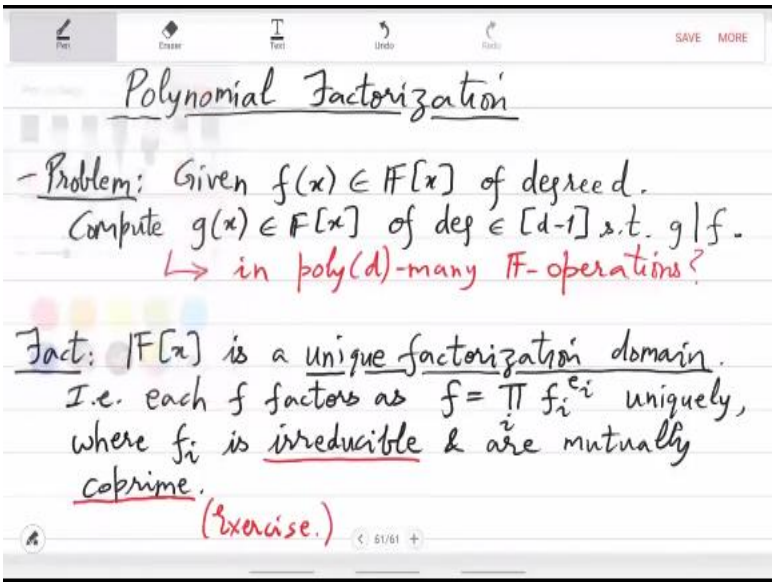
So if the tensor rank for n 0 is r 0 then what can you say about omega, the matrix multiplication exponential, right. So that will be log of r 0 base n 0. That is the exact relationship. So for T 2 you have shown 7. So it gave you log 7 to base 2. If for some bigger n 0 so you go to 10 by 10 matrices or hundred by hundred matrices and you come up with a magical algorithm or magical tensor decomposition with a very small r 0, right.

So then you may improve omega. This log of r 0 base n 0 may be less than the best known, okay. Yeah, that is about matrix multiplication. Any question? Okay. In other than Strassen's algorithm or the improvements on Strassen's algorithm have come not by directly working with this T and r but tensors related to this, okay. So T and r is still not understood. But there are some other tensors which are related to T and r.

And the tensor rank for those are bounded. And those bounds improve omega, okay. So there is a whole list of such complicated tensors but they are still, there are ways to study them in contrast to T and r. So that is what these matrix multiplication papers do. Yeah, but we will not study that. Okay, then we can move on to the new topic.

So I hope you have now enough experience in multiplying things, right be it numbers, polynomials, matrices. So now is the time to factorize, the opposite of multiplication.

**(Refer Slide Time: 51:07)**




And that is a real task. Okay, so we will start with the factorizing polynomials, because this is the simplest. Integer factorization is so hard that all your security is based on that currently. So we will start with first with polynomial factorization. So do you know any algorithms for polynomial factorization? Have you seen any ways to factorize polynomials? Or which special cases finding roots?

So what are the algorithms for finding roots? Yeah, so you must have seen only one algorithm which is Newton iteration or variants of it, right. But that gives you approximate roots. So what if you want to find exact roots? So we will actually not discuss for now. We will not discuss Newton iteration, we will discuss very different methods. Because we actually want exact roots, specially when you are over a finite field.

So if you are over a finite field, then Newton iteration actually, I mean, it is a symbolic computation algorithm. So you can implement it, but it loses its geometric meaning. Right? Because in a finite field, you cannot really you cannot really draw a genuine picture. It is a discrete domain, not a continuous domain. So we will start with the finite fields, we want exact algorithms, not approximation.

Yeah, it is Newton iteration, it works in many cases. I will not say it does not work. With moderated assumptions, it will work but the geometric pictures will all be false. So it will be the classic case of drawing wrong pictures, but getting correct algorithms. That also we will do in this course, but for now we will start with exact algorithms. So the exact problem is you are given a let us start with univariate polynomial over some field.

Field is F big F. This little f polynomial is of degree d. So you want to compute another polynomial gx in the same field of degree 1 to d – 1. Okay, so its degree should be less than that of F and it should at least be a linear polynomial. It should not be a constant such that g divides f. Okay, this is the question. So in simple terms, f is given and you want to find a non-trivial factor g.

A non-trivial means that it should not be a constant it should be at least linear. So degree one and it should not be the full thing so it should be degree d – 1. And we want to do this in we want practical algorithms for this. So which means that it should your algorithm should be in poly D many F operations. Okay, so the in terms of base field F operations, this your algorithm should be very fast.

So this is the exact factor finding. Why do we pick field? We only want to look at polynomial over a field because there you know that there is unique factorization, right. So unique factorization would in particular mean that the number of factors is that the number of irreducible factors is at most is at most d right. So d is in the input so you can basically output all the factors.

So you do not just have to find one factor, but once you have an algorithm to find one factor you can find all factors, right. So all these things are efficient then. So Fx is a unique factorization domain. So that is each Fx product of f i e i uniquely where f i's are irreducible. And e i's are just numbers. They are non-negative numbers and f i's are irreducible polynomials.

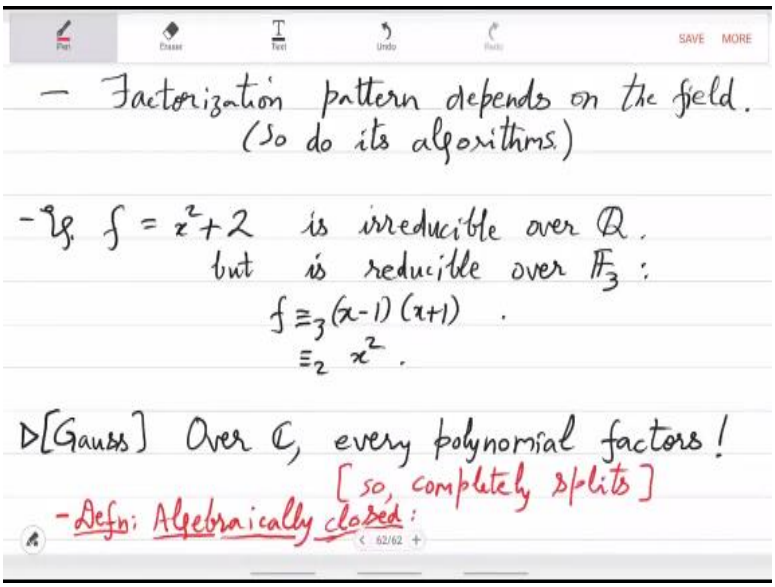
So this UFD means that F will always have this list of f i's will be unique and the list of e i's will also be unique. And if you compare the degree both sides then this means that f i's can at most be d many in fact sigma e i is at most d. So f i where f i is

irreducible and they are coprime. So they are mutually coprime. That is also important. Okay, any question? Yeah that is a fact.

Now you can actually prove it because you have seen GCD etc. Sorry. Oh everyone here may not no P id. Yeah, but by first principles, you can prove this which is use GCD algorithm. And so if there is a different factorization product g i e i prime then you can argue that g i has to be some f j. If it is not, if it is coprime to all the f i's then you can take GCD and so on.

So I will not go into the proof, but it can be proved by first principles. This is a pretty standard result. Division algorithm. What is an ED? Yeah, so it is ultimately Euclid GCD. Euclid GCD also used only division. So these are, all these things are related. So you can use any method, whatever you find comfortable in. Sorry. No, they may be proportional. One maybe f the other maybe 2f. Some unfortunate things.

**(Refer Slide Time: 1:00:11)**




So factorization pattern, of course, depends on the field. And so do factorization algorithms. They heavily depend on the field. So do its algorithms. Right? So which is why we will actually have several sub topics in this main topic of polynomial factorization. Each subtopic will look at the specifics of the field. Okay, so we will at least have 3, 4 parts of polynomial factorization.

So it is a long topic in this course. But the byproduct of all these sub topics will be huge. So the byproduct will have applications has applications in diverse areas in

computer science and in math. So example is that if you look at the polynomial x square plus 2 right, so over q it is reducible or irreducible? So why is it irreducible? Yeah, which is square root of –2.

So you have to show that square root of -2 is not in q, right? That is again a small proof. So do that proof. Why square root of 2 irrational. So by that proof, you know that x square plus 2 is irreducible. Over q it is irreducible, but if you go to other some other field like go to a finite field, say mod 3, then it will be? So over F 3, I will call it. It will factorize because there + 2 is – 1 and then it factorizes.

So x square plus 2 is the same as x - 1 times x + 1. In fact, it has two distinct roots. And mod 2 it is worse, because mod 2 it is just x square, so it has a repeated root, okay. So the multiplicity of the roots, where the roots are, how the factorization is, everything is dependent on the field. So do you know a field where every polynomial factorizes, right. So this is another fundamental theorem in, is it in algebra or analysis?

One of these two proven by Gauss. That over complex every polynomial factors. Do you know the proof of this? When I say polynomial, I mean univariate polynomial, right. We are only looking at univariate. So every polynomial factors, which means to what extent it will factor? It will factor all the way to degree 1, right. So in other words, every polynomial has a root and completely splits. That is called complete splitting.

And so C is called algebraically closed, okay. So this is the definition of algebraically closed field that any polynomial completely splits or degree d polynomial has d roots in the same field. Yeah, okay, so we will continue next time. Any questions? Have you seen the proof of this? Yes, but complex analysis is very heavy. So how short is the proof or how long is the proof?

The proof is actually very short. This is, the proof only requires this modulus operator. A complex number and you want to compute its norm basically. The proof is just uses norms, uses the norm and the relationship between coefficients of your polynomial and the roots, okay. So it is a very, it is actually a very simple proof, but

obviously is tricky and fundamental. So that also I leave as an exercise. So next time we will start factorization algorithms over finite fields.