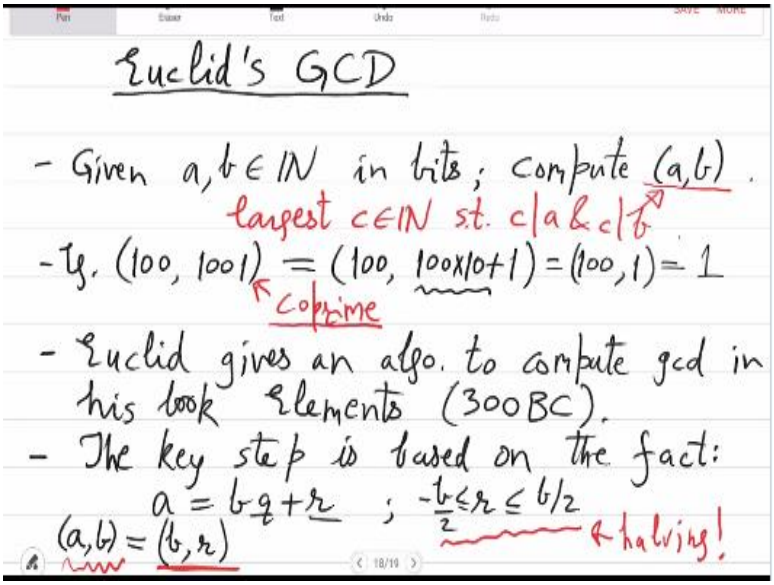
Computational Number Theory and Algebra
Prof. Nitin Saxena
Department of Computer Science and Engineering
Indian Institute of Technology-Kanpur

Lecture - 03
GCD Algorithm and Chinese Remainder Theorem

(Refer Slide Time: 00:13)



So we started Euclid's GCD. So you are given two integers a and b in bits. And you want to compute the largest c that divides both. So if the largest c is 1, then we call the numbers to be coprime. Otherwise we say that they share a factor, right. So like 100 and 1000 share a factor but 100 and 1001 they do not. So they are coprime. That is the definition of coprime.

So Euclid gives an algorithm, gave an algorithm to compute GCD, which we call Euclid's GCD algorithm. And this is based on long division. So it will basically reduce the a, b this computation. It will reduce this computation to some smaller instance of the same problem, right. So it is basically a recursive algorithm which is how you must have implemented it in ESC 101.

So you assume that a is greater than b. And you can also assume that both of them are positive. And so this is greater than 0. And then you compute the remainder of a mod b. So that remainder is r. So you instead of working with a now you will work with r,

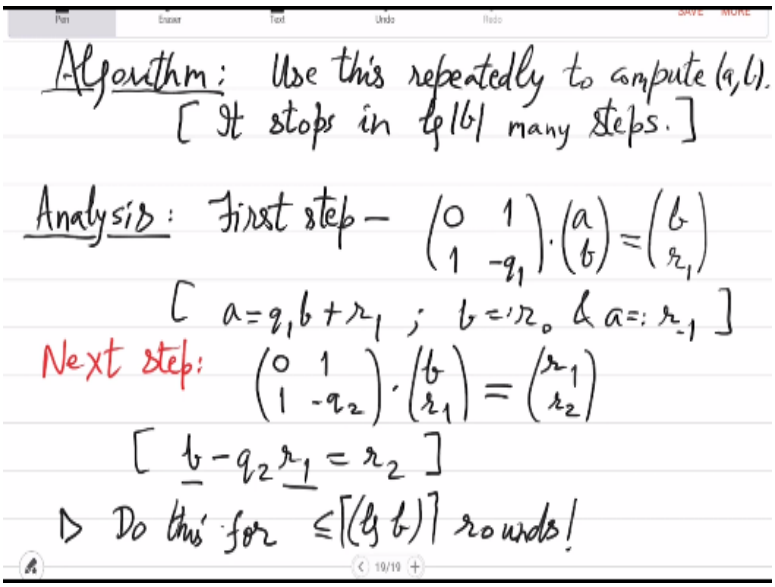
okay. So in this case now, you can either say that the remainder is between 0 to b – 1. That is one way, but we will prefer this.

We will prefer that the remainder be b by 2 or less. So then the issue with this is that what to do with b by 2 to b - 1 range. So that range has to be reflected with negative sign, right. So you subtract b, so that is the left end - b by 2. So now we can truthfully say that the instance before has been halved, right. So previously, the instance smallest number was b. In the new instance, the smallest is b by 2 or less. Right.

So in this sense, there is a halving that has happened. So that is the key idea that you halve the instance. And so you can see that this process when continued, will very quickly converge to the GCD, right. How quick how fast? How many steps are there? Log b, right because the smallest one is being halved. So it does not matter what a was, only depends on b.

And then log b many steps it will be reduced. It will be reduced to basically 0. You will get 0 and then what if one number comma zero will be the GCD. That is the last base case. Yes. Yeah. So I am being inconsistent. That is true. So let us remove that assumption. Some students will not like inconsistency.

(Refer Slide Time: 04:03)



Okay, so the algorithm is just, use this repeatedly to compute the GCD. So it will stop. So now since b can also be negative, let us remove the sine and then take log. So it stops in log b many steps. So you can think of the unsigned value of a to be more than

the unsigned value of b. So b is still smaller in a way and that many steps you will require. So let us do the formal analysis because that I think very few would have seen.

So this algorithm what exactly is the time complexity? Exactly I mean in terms of asymptotics. So number of bits of a bits of b what is the time complexity analysis. If you do this lazily then you will get a bad bound. So we have to do this a bit more carefully. So let us see this as two by two matrix transformations okay. So you started with a vector a, b and you are reducing it to b, r vector.

So there is a matrix transformation happening. So what is that matrix transformation? So first step is **is** the matrix 0 1, 1 –q 1 times a b. So a times 0 plus b gives you b and a -b q 1. So q 1 is the previous in the previous slide the quotient that you got. You are not now calling it q 1. So a – q 1 b will be the remainder. So now let us call it r 1. Okay, so here this is the equation a – q 1 b or you divide by b and then get the quotient and the remainder.

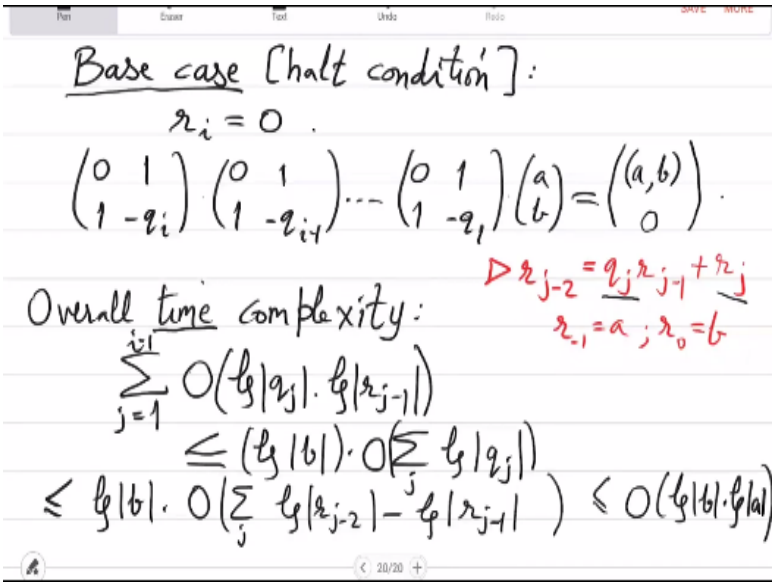
And in the previous step you had a, b so we can call b r 0. And we can call a r -1. So that will be consistent with the next steps. So we started with r -1, r 0 and then we get to r 0, r 1. Okay, that is we move to the next step. So the next step is now, so the matrix will be of a similar type in the next step also okay. So it will be 0 1 and 1 – q 2. So times b r 1.

Because we have ensured that in the first step b is somewhat smaller than a and then in the next step r 1 is somewhat smaller than b. So similar kind of transformation will give you r 1, r 2. Okay, so here the equation is well, the first row gives you r 1 that is clear. The second row is the long division. So b is equal to let me write it in the same way. So –q 2 r 1 is equal to r 2 okay.

So you divide b by r 1 and that will give you the quotient q 2 and the remainder r 2. Remainder r 2 again you will assume is around half of r 1 in magnitude, right. So the algorithm thus proceeds. So it will have log b many steps. So let us note that. So do this for log b rounds. So this lg will always denote log to base 2, right. In this case is the number of bits in b.

Maybe you need to take a ceiling here. So do it for these many rounds and then what happens? So then you will get to let us say r i and are r i + 1 where r i + 1 has become 0 right. So at some point it has to go below 1, but below 1 the only integer possible is 0 in this case. So at that point then the GCD will become trivial to calculate because you are looking at some integer r i and the integer 0. So that is the base case. Right so let us write that down.

(Refer Slide Time: 10:05)



So the base case will be or the halt condition. So base case of the recursion or in this iterative implementation is the halt condition. So this is when r i is equal to 0 okay. So this is you have these many transformations on a, b. So now you can see that the every step is just multiplying by a two by two matrix of this kind of triangular form, it is not exactly triangular, but triangular with respect to the anti-diagonal.

So you can compound it and multiply by all these two by two matrices in one shot to the vector a, b and that gives you 0 here and what is the second element? So this is the GCD right. So you have computed the GCD of a, b in the after i steps. So what is the overall time complexity? So remember time is counted in terms of bit operations, right. This we had said before, that is the definition of time.

So you have to do the analysis and then in the end give a function of log a and log b. So let us do it step by step. So for the jth step what do you get? So j is equal to 1 to i or maybe i -1. So what is the jth step complexity? What should I put here? Log of q, j

and log of right. No, that was not the, so this is just the complexity of long division. So in the jth step, what is the complexity of long division that we saw in the last class?

Yes, so yeah, I think you are right. So let me write down the invariant first. So the invariant is r j -2 is equal to q j plus r j. This is the invariant, okay. So you divide the r j -2 by r j -1 getting the next remainder r j and remember that r -1 is a and r 0 is b. So these are the equations you have, right. Is this correct? So j = 1 will give you a = q 1 times p + r 1 and so on, right. So sanity check is fine.

So looking at this, so what is the complexity of computing here q j and r j? That is the question. That so that complexity you have to put inside the bracket. Right? So this complexity is last time we saw it is log q j times log r j - 1. Yeah. So this is the sum we are interested in. Is this any questions here? So I am being very careful here.

Because if you are not careful, so one lazy analysis that you can do is just put here log a times log b, right or just put here log a square, log a whole square. But then the sum will give you another factor of log b. So you will actually get essentially a cubic time complexity. But if you do my way, then you will get quadratic complexity, which is the real complexity of this algorithm.

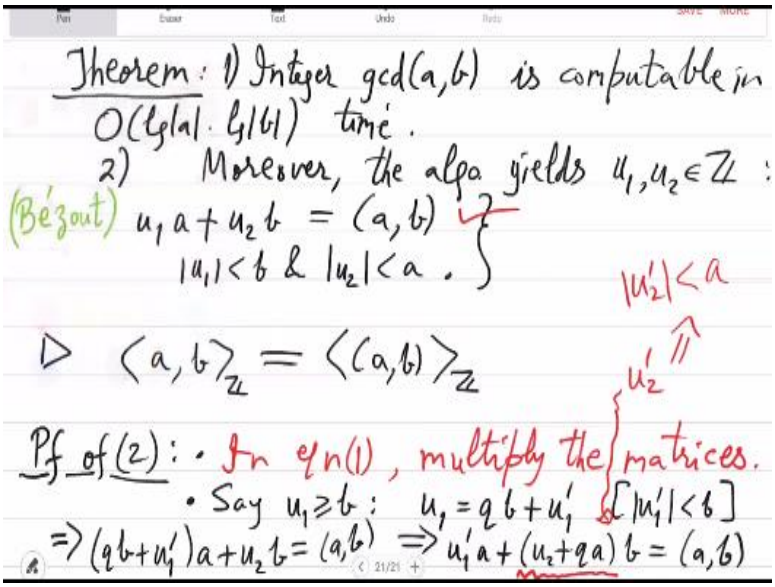
So it is not cubic time, it is quadratic time. So how do you get that from this sum? How do you simplify this? So this is less than equal to, we cannot analyze this sum of product. So we have to remove something from this product outside and upper bound it, right. So we can remove from here. Maybe this log of r j -1 we remove. So r j -1 is at most, at most b exactly. For j = 1, it is just b.

So that is what we can bring out. And now we get some space to do further simplification. So this is now only sum of, so let us bring the big O outside. That is allowed, right? So now you only have to analyze sigma log q j. So what is that? Again, if you are not careful, then this will become quadratic, this thing inside big O because the sum has already log b many terms.

So you will get log b times log a, you do not want that. You want to do this carefully. Any ideas? Look at the invariant equation. Invariant equation tells you that q j, so q j times r j -1 is equal to r j -2 essentially. So when you take log, log q j is kind of the difference between log r j -2 and log r j -1, right. So you can replace this by the difference. So r j -2 minus r j -1. Is that fine?

In this equation r j is very small. So you can say that log q j is at most, it asymptotically it is just difference of these two logs. And now what? Yeah then you can do the telescoping sum. So you can see that everything will cancel out except one thing which is log a. So you get log b times log a. Is that fine? Right. So this analysis tells you that Euclid's GCD algorithm is actually quadratic time. And if you do this lazily then it will give you cubic time.

(Refer Slide Time: 18:25)



So what we have shown so that is your first theorem in this course. And it is a very old theorem that integer GCD is computable in log a times log b time. So that is one thing you learn, but this algorithm actually gives you more right. So that also you must have seen that it actually gives you a combination of a b that is equal to GCD right which is called extended Euclid GCD algorithm.

So how does that work? So I want to claim here that moreover, the algorithm yields u 1, u 2 that are natural numbers such that you want u 1 a + u 2 b is equal to the GCD. One of them has to be negative. Okay, probably I can put a minus sign here then, but

let it be symmetric. So magnitude of u 1 is smaller than b and the magnitude of u 2 is smaller than a okay.

So this algorithm actually gives you this combination linear combination of a b, if you like, where both the coefficients are under control. So u 1 is smaller than b and u 2 is smaller than a. So the opposites kind of such that the linear combination is exactly GCD, right. This is something which could not have been predicted by the definition of GCD. So this is a rather special and surprising property.

So what this tells you in the terms of ideal? So if you look at the ideal of a and b what is it? Ideal over integers. Yeah, so it tells you that this ideal which is apriori generated by two generators actually has a single generator which is the GCD right which is quite unexpected. So this starts the theory of principle ideal domains and so on or unique factorization domains which you have must have studied in algebra courses.

So this is saying that if you take an ideal of numbers, then it is generated by only one number and that one number is the GCD of those numbers. So why do you get that because, obviously both a and b are in the RHS right? Because GCD divides both a and b. So they are by definition of ideal in RHS. Now why is the RHS contained in LHS? That is the identity above, right.

So by this identity yeah, do not rush. Let us first understand this. So that first equation is telling you that a, b can be expressed as a combination of a b but that combination is actually a Z combination and hence RHS is an LHS. So LHS is equal to RHS right. This is what you have gotten. But what is the proof of the theorem especially the second part? So what is the proof of 2?

Do you see that the previous slide gives that? Yes, right. So you just focus on this equation. So you look at this equation 1 and what will you get if you multiply these two by two matrices? So you will get a single two by two matrix. And let us say it has elements alpha, beta, gamma, delta. So then alpha, beta with a b inner product is equal to GCD. That is the simplest proof immediately follows.

So in equation 1, multiply the matrices. And the second part of second part that u 1, u 2 are small, how do you get that? Why are they small? So that is actually very simple. I think you see this in discrete math or some courses. So suppose u 1 is not small, suppose it is b or more. Then you divide it by b get the remainder and you rearrange this equation.

So you can write, so say u 1 is greater than equal to b in magnitude but we can also assume u 1 is actually greater than equal to b, then what you can do is you remain compute the remainder q b + u 1, u 1 prime let us say where u 1 prime is small. So now the equation will be q b + u 1 prime a + u 2 b is equal to the GCD. So u 1 prime a is what you want to be there, right; q b a is the garbage.

So what do you do with this q b a? Well you absorb it in the subsequent term. So you will get u 1 prime a + u 2 + q a b is equal to GCD. Now does it help? So this is now, so now u 1 prime is under control it is below b. But what can you say about this second coefficient u 2 + q a? What do you think? So the idea is simple. This u 1 prime a, how big is this product?

In magnitude it is at most a b, I mean it is smaller than a b in fact. The GCD is much smaller than a b, so that we can even ignore. So this equation is saying that something which is smaller than a b plus the underlying coefficient times b. They are roughly they are basically equal, roughly equal. So u 1 prime a and this u 2 plus q a we can call it actually u 2 prime.

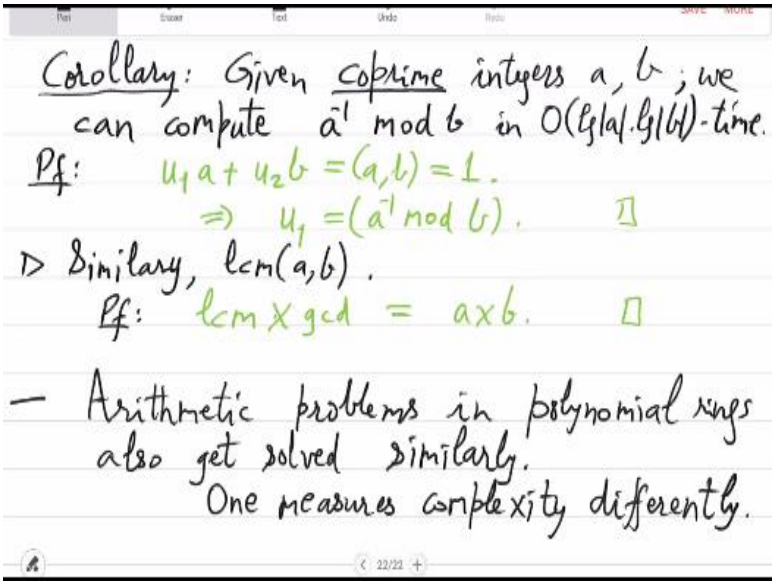
Let us call it u 2 prime. So u 1 prime a and u 2 prime b they are roughly equal in magnitude. That is what this equation is saying. Which means that u 2 prime is at most, it is at most a right. If u 1 prime a is at most a b and these two are roughly equal then u 2 prime cannot exceed a, okay. So you deduce that this is smaller than a. So that is your final equation. That is what you will output.

You will output u 1 prime and u 2 prime. Is that clear? So in fact the way algorithm is written this step might be needed. So u 1 may blow up, but then you reduce it and when you reduce it, then the corresponding u 2 prime is also reduced. So both are

reduced simultaneously okay. That is another amazing property of this identity. So this identity is also called, is it called Bezout identity?

Yeah, so that is a that this identity has a name. So that is the Bezout identity Any questions? Okay. So now this algorithm is extremely important. We will see the importance again and again in this course. But for now, let us just write these corollaries.

(Refer Slide Time: 28:34)



So given coprime integers a, b we can compute a inverse mod b in the same time. How do you do that? How can you invert numbers? So first of all, why should a inverse exist? Right but why should it? So in fact, even the reason why a inverse will exist goes through Euclid's GCD right. So Euclid's GCD algorithm tells you that a inverse mod b exists and not only it exists it can be found efficiently, okay.

So 1 over a can be found mod b. Why is that or how is that? Yeah, so you have this u 1 a + u 2 b = GCD of a, b which is 1. So now you look at this equation mod b and u 1 is your answer. So u 1 is a inverse, right. So that is all. So that is the significance of u 1, u 2. For coprime numbers, u 1 is essentially the inverse of a and u 2 is inverse of b, right. So inverse computation is made simple now.

And similarly, you can compute LCM. So how do you compute LCM using GCD algorithm? Yes, so this is LCM is equal to or LCM times GCD is equal to is equal to the product, right. So you find one you find the other. So use this property. First prove

this property and then this will give you the algorithm for LCM also. So LCM is the number that is divisible by both a and b.

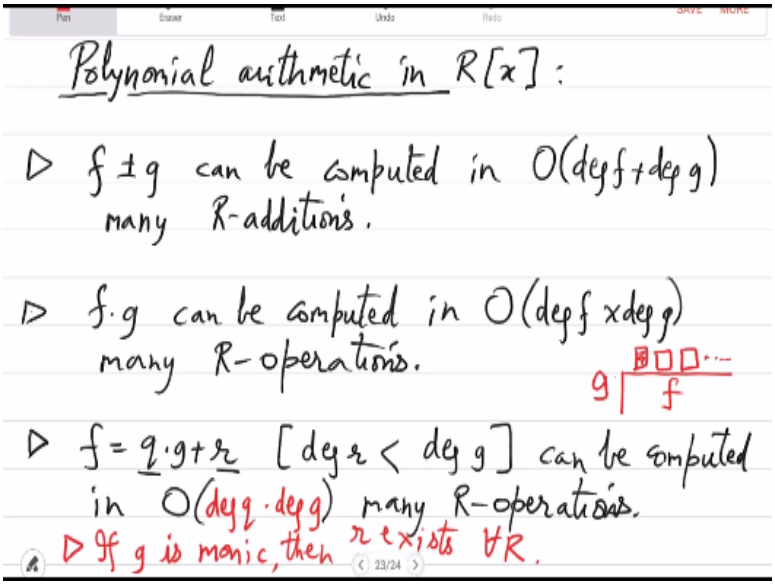
It is the thing opposite to GCD in a way, is the least common multiple. So that also you can compute while the definition of LCM seem to need factorization of a, b right. But this fast algorithm is completely factorization free. That has to be the case otherwise, you will be stuck with factorization. There is no known algorithm for factorization. Okay.

So now once you have set up this machinery, you can now apply it in very different rings, okay. So this computation you did in the integer ring, but you can do the same as was pointed out before. You can do this for polynomial ring as well. But then the complexity metric will change. So we will see how it changes. So arithmetic complexity in polynomial rings also gets solved similarly.

But now one has to properly measure the complexity or one measures complexity differently. So that we will see now. So in a polynomial with integer coefficients, how will it be presented in the input? It will be presented coefficient by coefficient which are all in bits. And how many will they be? How many coefficients will there be? This is around the degree d + 1.

If degree is d then d + 1 many coefficients. So this d will also enter in the complexity So it is not only the bits that a coefficient needs, but it is also d's. It is kind of bits. So log of a coefficient times d, right. That is the new input complexity and also the output complexity is similar. So you have to now state the complexity in those terms. So let us see that.

(Refer Slide Time: 34:28)



So polynomial arithmetic in R x. So R is a ring with where you know how to do the ring operations, addition, multiplication. In fact for this discussion we will consider it to be unit time. So we will not worry about the ring operations in R. We will only worry about once given these sub routines to add and multiply in R what can you do with the polynomials? How much is the complexity of the polynomials?

You can think of R as Z. So f plus minus g, f and g both are in R x. So this sum or difference can be computed in how much time? So not time sorry. We cannot say time here. We will just say how many ring R- operations will it take to compute the sum. So this will only take degree of f plus degree of g many R- additions in fact, not even multiplication. So instead of time now, we are moving to a bit more abstract measure.

We are not worried about how much time R- additions will take. That somebody else will solve. But once it is sold, how much will the f + g computation take? So that is bounded by the degree. Now what can you say about multiplication? How many R-operations? Yes, so this will be multiplicative right because if you look at the definition of polynomial multiplication, there is this convolution that happens to compute the coefficient of x to the i.

You have to see in how many ways can x to the i be formed. So there is you pick x to the j from f and you pick x to the i minus j from f and then do this for all j right. So actually to find even the coefficient of x to the i, you have to do a linear scan and then

there are degree of f plus degree of g many coefficients. So it becomes quadratic. So this is degree of f times degree of g. Is this clear?

So this is just formalization of the high school algorithms that you know, right. For polynomial arithmetic these are the algorithms you already know. But this is the time complexity. So now let us make it more complicated and talk about division. So f equal to q times g + r. So you have to find the quotient and the r which are now polynomials in x such that degree of r is so degree of r has to be less than the divisor degree which is g, right.

So this can be computed in how many ring operations? So what is this? So you have to go back to Euclid's GCD algorithm and whenever you are doing the division step you have to now do polynomial division, right and so well what is the complexity? Actually this is not GCD yet so we are only talking about one step which is division. So look at long division so how many R- operations would be needed there?

Yeah, why is that? So you will have g and f and then here you will be computing the coefficients basically one by one right? So the first monomial that you will compute this one, this monomial will be the leading monomial of f divided by the leading monomial of g, right. So that is the leading monomial and the coefficient will be similar.

The quotient, the ratio of the coefficients corresponding coefficients and so on. So you basically compute one monomial then you multiply it by g then you subtract that from f and repeat, right. This is the long division process. So how many steps would it have? So how many square boxes are there? So degree of f minus degree of g right. Or in fact in terms of this degree of q. So degree of q many steps are there.

And in each step, how much price are you paying? Degree of g. So that is the need complexity. It is now completely analogous to what you got for numbers. So there you got log of q, here you have degree of q right. So generally this would be the dictionary. For numbers log the bit size and for polynomials the degree. That is how this conversion happens? Yeah. Yeah, that is true.

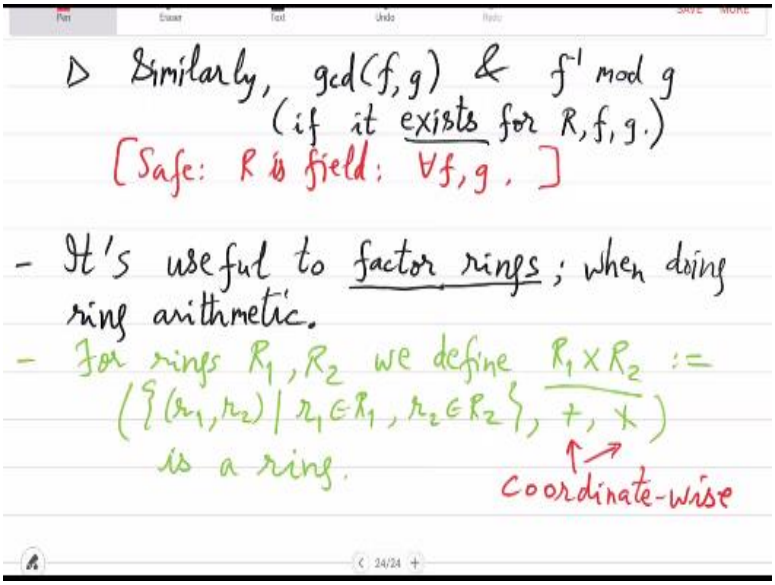
But one can say that either this algorithm will give you the output or it will get stuck because it tried to divide by 0. So the program may give division by zero error, we do not care. So at any point if it finds a zero divisor sorry? No, in one step what are you doing? You are dividing the coefficient of f with coefficient of g. So as long as that division is defined, the algorithm will continue.

It is actually an important point. So if the coefficient of g was 1, so which means that g was monic then does not matter what the ring is, right? So maybe that one can remember. So that is an important point. So if g is monic, so yeah, the question really is, why should q and r or why should r exist? So if g is monic, then r always exists for all rings R, okay. So this is a very useful fact that generally you do not have to worry about r being a field.

It can be any ring because usually in applications we will make sure that g is monic. So if g is monic, then it does not matter, division by zero will never happen. If g was not monic, then there may be a silly coefficient sitting which may behave like zero in that ring. And so when you ask the computer to divide 1 by 0, it will complain and the program will crash. So that is fine.

If it does not crash, then it will take these many steps, degree of q times degree of g. In fact, this also is a nice mathematical fact that r exists if g is monic irrespective of what ring R is. It does not depend on field properties, okay. So now with armed with long division you can talk about GCD, right. GCD is now also within reach.

(Refer Slide Time: 44:22)



So similarly GCD of f, g. So why GCD of f, g? In the same time degree of f times degree of g right. So there is a important difference in the proof. So in the Euclid GCD proof that you saw for integers we were saying that the instance the number is being halved. Now is something being halved here. So here actually nothing is being halved. Here actually just the degree is being reduced by 1.

Because the division step tells you that degree of r is less than degree of g. It is not half. So it will actually take degree of g many steps. And then when you couple the proof with this fact, the previous proof with this fact and you sum it up, you will still get degree of f times degree of g. Okay, so you can do that as an exercise. Is the same proof but there is a key halving step missing.

Yes. Yeah, exactly. So we were doing halving there because the measure was log of the number. So when you halve the number then log changes by -1. Here degree will change by -1. So in that sense it is analogous. **"Professor - student conversation starts"** GCD may not exist because r is not monic. Since r is not monic so the next step you cannot have like same algorithm if r is not monic. **"Professor - student conversation ends".**

Yeah, but if it does not exist, the program crashes. Do not worry. Yes, I should add here that if it exists. Similarly GCD and f inverse if they exist for R, f and g. So you may find examples of the ring and two polynomials when GCD does not exist, okay.

If r is a field, then it will always exist. If r is not a field, then some step may get into division by zero. But not for every f, g.

So for ring R, that is not a field there might be good instances of f, g. So whatever is given to you in the input the algorithm will accordingly decide. Okay the algorithm will not prejudge the input. Any question? You may be theoretically right. But if it exists maybe this will find it. Try to prove that. Yeah try to prove that if it exists then this will find it. Yeah, so to be safe so what is the safe setting maybe I should also write that.

So the safe setting here is R is a field and then does not matter what f and g is. If R is a ring so non-field then safe is g is monic. Well not for GCD. So then we do not know what safe is. So just erase this. No when you get to the remainder after few steps it is out of control so let us remove this. So for field this makes sense for other rings it may crash. Sorry. Yeah. Right.

So next thing will be, it is to do with factorization of rings. So if you have a ring or say to put it simply if you have a number n, which is a product of two primes, like two times three, right? So the arithmetic modulo 6, is it related to arithmetic mod 2 or mod 3 right? So is there a connection between the arithmetic of ring and a sub ring kind of. So number and a factor? What is the connections we will formalize that.

So that thing will be true quite generally, but we will first give it for integers. So it is useful to factor rings when doing ring arithmetic. So because then the arithmetic can be reduced to simpler rings, rings which cannot be factored, which do not have factors in decomposable rings. So if you have not seen this product of rings then let us define that.

So for rings R 1, R 2 with completely different operations possibly, addition multiplication is completely different in these rings. The sets are completely different. There are these there are many ring constructions that you can do. Using a ring and especially using two rings you can construct new rings. So one such construction that is relevant here is R 1 cross R 2.

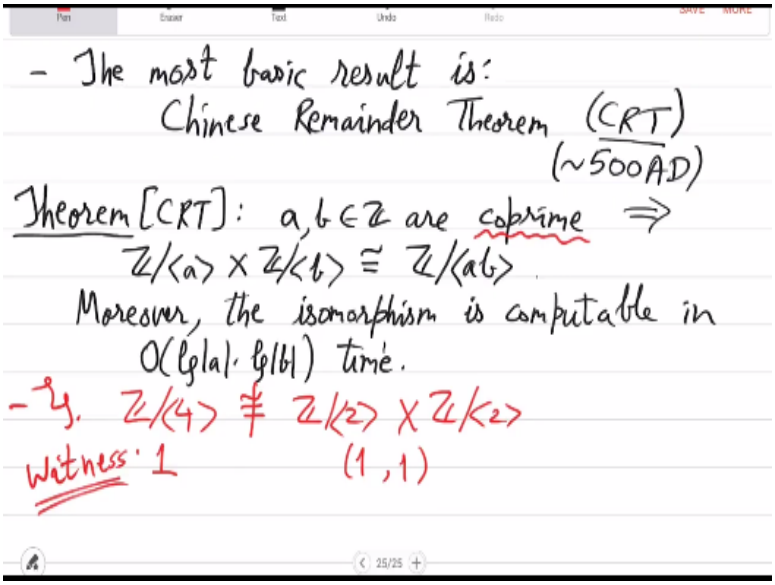
So what is this ring? So as I said this ring is just pairs r 1 from R 1 and small r 1 or r 2 from big R 2. What are the addition and multiplication operations? Yeah, so the multiplication and addition is coordinate wise, right? So in the sphere r 1 is in a different world and r 2 is in a different world. So if you want to add r 1, r 2 with r 1 prime r 2 prime, then you can just add r 1 and r 1 prime and r 2 with r 2 prime and get the new pair.

And the same thing you can do with multiplication and you can convince yourself that this is a ring again. No completely different r 1, r 2, no relationship; r 1, r 2 are independent. No, when I say that r 1 cross r 2 is a ring, I have to define the operations for the set elements. The set element is a pair. This is a third operation yeah; r 1 had one r 2 had second and this product has third.

So this is also called I think, direct product of Turings or external product. Yeah, but we will simply call it product of rings. Factor of r 1 + r 2 we will call r 1 or r 2 okay. So we see this as something that we saw in integers. That is our notion of factorization of rings or product of rings. So this is coordinate wise okay. So now the question is, if you have a ring R, whose factors we know are r 1, r 2.

So r is essentially or r is isomorphic to r 1 cross r 2. Then doing ring arithmetic over r how is it related to ring arithmetic over r 1 and over r 2, right. That is the fundamental question. So the most basic result here is what is the result called? Sorry. Yeah, that is fancier, something more basic. Yeah, so Chinese Remainder Theorem.

(Refer Slide Time: 53:57)



And remember we are interested mainly in algorithms. So whatever we say about structure theorem, we have to show that it is efficient. It could be very constructive. So the most basic result is Chinese Remainder Theorem. So we will be using this all the time in various forms in this course. We will call it just CRT. Okay this is an old result, possibly in different form.

So around 1500 years old. We will just see the proof in terms of integers and then you can generalize it accordingly. So the theorem for integers is so a, b are coprime integers. Then how do you see integers mod ab? So this ring given by this mod ab arithmetic, how is it related to mod a versus mod b? Yeah, so that is what we claim that this is actually isomorphic to doing things in parallel.

So one of you does computation mod a and your friend does computation in mod b. And then you combine these two results to get the answer mod ab. So that is the existential thing, but it is actually very constructive. So moreover, the isomorphism is computable in log a times log b time. Okay, so that is the interesting part for us that this isomorphism the ring, mod ab is actually strongly related to mod a and mod b and this connection can be used in quadratic time.

So for example, Z mod 6 is factors as Z mod 2 cross Z mod 3. Why did we use coprime? Is there any reason? Why this extra assumption? So let us see an example. So Z mod 4 and Z mod 2 two times. So is there a relationship between z mod 4 and

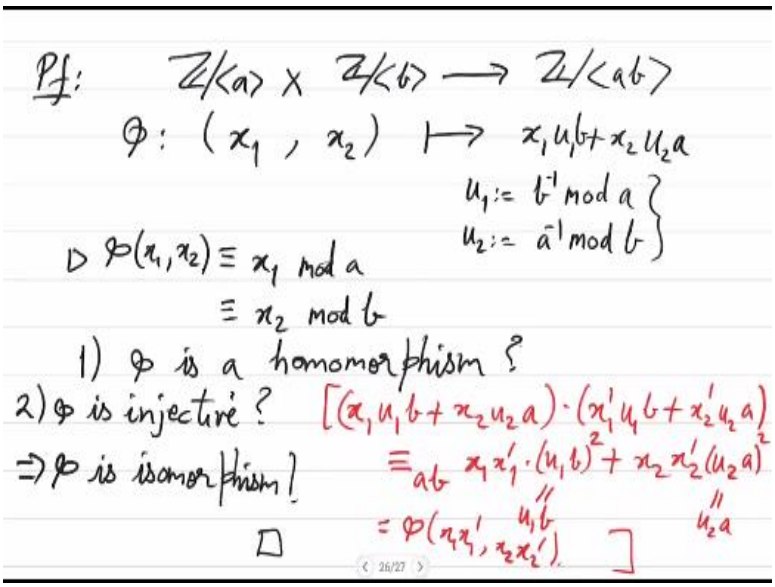
the RHS. Why are they not isomorphic? Interesting. Yeah so cyclic group with respect to what? Two operations are there right?

Yeah, but with respect to what operation are you talking about? Addition, right? So if you just look at the additive structure on both sides, you will see a difference, okay. You do not even need to go to multiplication. So if you look at the element 1 so you know that unity or the identity element in the LHS is 1, the identity element in the RHS is 1, 1. So if it was isomorphic, they were isomorphic, then these have to be the same.

But if you make them same, then the problem is the 1, 1 if you add it to itself, you get 0. From the LHS you do not get 0, you get 2, which is not 0 right? So that is the contradiction. So that is the witness in fact. This is the witness for non-isomorphism. So coprimality is necessary. Without coprimality it is always false. Okay there is no saving grace okay. So we have to use coprimality in the proof right?

How will you use it? Yeah inverse and Euclid GCD. So GCD will give you the inverse. It will exist when the numbers are coprime. That is what will be used in the proof.

(Refer Slide Time: 59:51)



So let us, so to give a isomorphism first you have to give what? A homomorphism right? First you design a homomorphism and then you lazily claim that it is an isomorphism right. So what is this first step? What is the homomorphism? So suppose

you have u, v element in the LHS ring. So what should you associate u, v with in the RHS ring? Easier to give yeah, that will be sure.

But what I want to do you will have to do anyways, you cannot escape it. So but you can take a hint from RHS to LHS. So basically you want a number in the RHS which mod a takes you to x 1 and mod b takes you to x 2. So what is that magic number in the RHS? Okay, so the map will be this x 1 u 1 + x 2 u 2 such that mod a it should become x 1.

So what we will do is we will take u 1 to be b inverse mod a that is true and we will take u 2 to be a inverse mod b. And of course, we can compute u 1, u 2 efficiently. In quadratic time you compute u 1, u 2 and then look at the linear combination of x 1, x 2 like this right. So when you take remainder mod a you will see that you will get x 1 because u 1, v becomes 1.

And when you go mod b you will see that you get x 2 because u 2 and a are inverses right. So these are very these are independent moduli. So mod a and mod b have nothing to do with each other, right? But this single number is able to satisfy both the constraints. So it is a clever construction and so that is your map phi. Now why is this map a homomorphism?

So well first property we have observed is phi x 1, x 2 is x 1 mod a and it is x 2 mod b. And the other properties are first is that phi is a ring homomorphism. Why is that? Why is it a homomorphism? So you have to see what how does it behave on addition? It is clear that I mean since the image is a is linear in both x 1 and x 2 it will behave well with addition, right?

Now what about multiplication? So x 1, x 2 times x 1 prime x 2 prime what will happen there? So maybe that you see. So what is this? And this is you are doing mod ab. This multiplication you do mod ab. So what will this give you? Exactly, so the fortunately the cross terms are zero in this ring. So this is only multiplying x 1 with x 1 prime. Let me write that u 1 b square.

So first we get this what is u 1 b square? That we have to understand, mod ab. Do you see that it is u 1 b? So show that this is actually u 1 b. And this is u 2 a okay. So the square actually has no effect. So you so this expression is actually equal to x 1 x 1 prime one coefficient and the other coefficient is x 2 x 2 prime. So which then is actually the image of phi x 1 x 1 prime comma x 2 x 2 prime.

Right that is the magic. So this is why multiplication is also preserved. Because the phi of this x 1 x 1 prime x 2 x 2 prime is actually phi of product. So phi of product is equal to product of phi. So this is why phi is a homomorphism. Second property I want to emphasize is phi is injective. Why injective? If it was not injective then something must be mapped to 0 right?

So 0 means that this x 1 u 1 b + x 2 u 2 a will become a multiple of ab. It will be zero mod ab. So what will that give you? You can actually deduce from that that x 1 and x 2 both have to be 0. So the only preimage of 0 is 0. So phi is injective. Is that enough for isomorphism? Do you need to prove that or can we end the class? Exactly, so since the both are finite rings, we do not need to go there.

It is automatically subjective, okay. So which means that this is actually an isomorphism, right. So that is the full proof. Any questions? These basic ring calculations you should fill in. It will be good for your education okay. But overall this is how you prove these things. And every step here is efficient. So from the RHS you can go to LHS and from the LHS you can go to RHS, okay.

Both sides, so phi and phi inverse both are efficiently computable maps, right. So this is completely constructive using Euclid's GCD. Okay, stop.