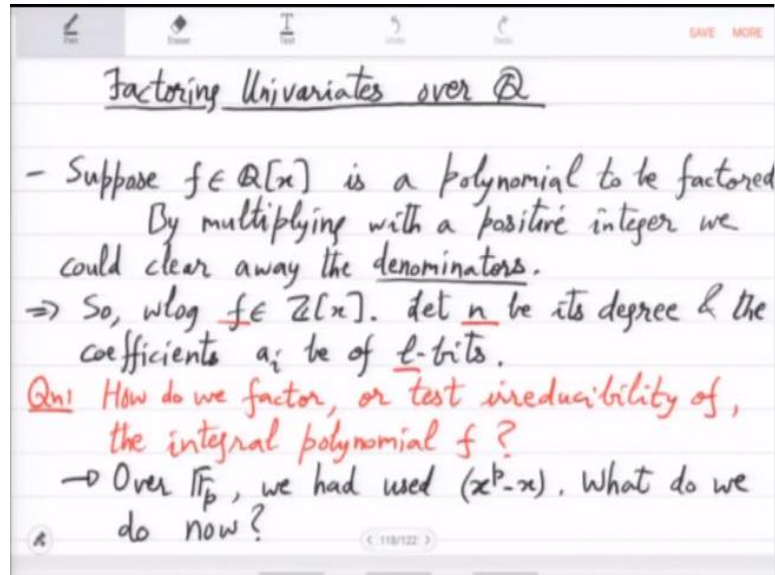


Computational Number Theory and Algebra
Prof. Nitin Saxena
Department of Computer Science and Engineering
Indian Institute of Technology-Kanpur

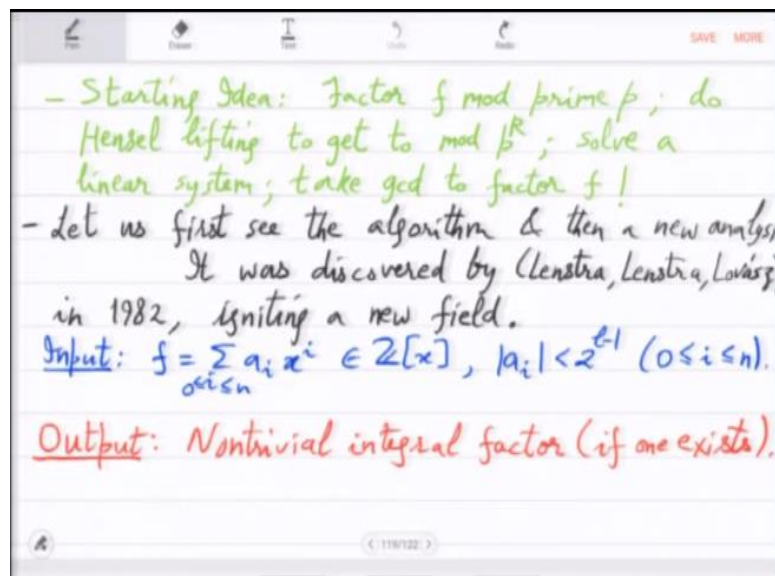
Lecture – 20

(Refer Slide Time: 00:19)



Okay, so last time we started this algorithm that is going to factor univariate polynomials over rationals.

(Refer Slide Time: 00:26)

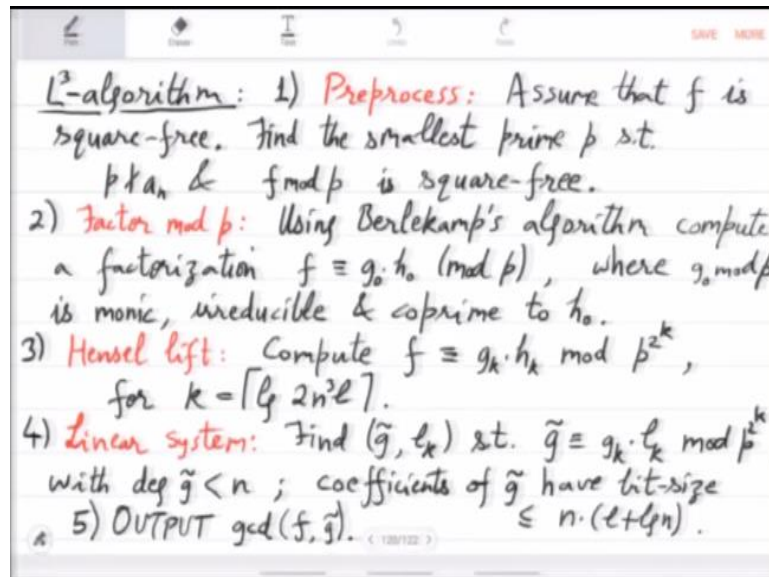


So basically in the input what you are given is f with coefficients a_i 's which are integers, you can assume them to be integers. And say these integers are given in binary l bits, okay. So you are given $n + 1$ integers, each is l bits. So we are assuming

here that these integers are sine integers and their value is smaller than 2 raised to $1 - \frac{1}{n}$. The in the output of course, we want to check whether the polynomial f is an irreducible polynomial.

If not then you have to output n integral factor, non-trivial factor. So the degree should be between 1 and $n - 1$ of this non-trivial factor, okay.

(Refer Slide Time: 01:21)



So we already actually wrote this algorithm, which is called L cube algorithm. So in step one, it is a basic preprocessing step. The idea is that you pick a prime p and then you factorize f modulo p . What we want, the condition we want on the prime p is that it should not divide the leading coefficient a_n . So a_n should be nonzero mod p . And $f \bmod p$ should be square-free. In the second step, this is simply a factorization step.

So you factorize $f \bmod p$. p will be a small prime. So it is enough actually to use Berlekamp algorithm, okay. So this will be a deterministic way to factorize $f \bmod p$. So let us say the factors are g_0, h_0 ; g_0 is assumed to be monic, irreducible and coprime to h_0 . It is a coprime factorization. Next is Hensel lifting. Okay, so now you lift g_0 to g_k , okay. g_0 is this monic irreducible coprime factor.

So at this point, you can apply Hensel lifting lemma constructively. And you can lift this factorization mod p raised to 2 raised to k . So it applied k times. k is sufficiently large. So we will see why this parameter is important, but it will be around \log of $n!$

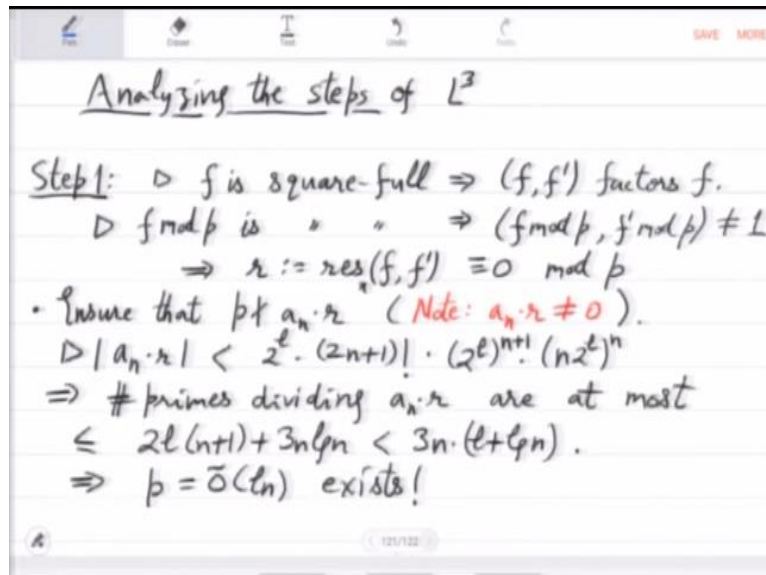
cube 1, okay. Fourth step will be a, will be the step where we will go from g_k to a polynomial \tilde{g} , which is closer to a factor, actual factor.

So remember that g_k is a factor of f only mod p raised to 2 raised to k , right. From here we have to go to an actual factor integral factor of f . So this is reminiscent of what we did in multivariate factorization before, right when we reduced bivariate factorization to univariate factorization, we did a similar linear system step. So here we want to find \tilde{g} and l_k cases that \tilde{g} is a multiple of g_k by l_k .

Obviously, degree of \tilde{g} should be between 1 and $n - 1$. Moreover, the coefficients of \tilde{g} should not be too large. So the coefficients of \tilde{g} should have bit size around the same as before which was 1 . So now we want it to be around $1/n$, 1 times n , okay. This much leeway we give, but not bigger. And finally, like we did in bivariate factorization, you take the GCD of f and \tilde{g} and output it, okay.

So this is reminiscent of bivariate factoring. We had also started analyzing the steps.

(Refer Slide Time: 04:36)



Analyzing the steps of L^3

Step 1: $\triangleright f$ is square-full $\Rightarrow (f, f')$ factors f .
 $\triangleright f \bmod p$ is " " $\Rightarrow (f \bmod p, f' \bmod p) \neq 1$
 $\Rightarrow r := \text{res}_x(f, f') \equiv 0 \bmod p$
 • Ensure that $p \nmid a_n \cdot r$ (Note: $a_n \cdot r \neq 0$).
 $\triangleright |a_n \cdot r| < 2^l \cdot (2n+1)! \cdot (2^l)^{n+1} \cdot (n2^l)^n$
 \Rightarrow #primes dividing $a_n \cdot r$ are at most
 $\leq 2l(n+1) + 3n \lg n < 3n \cdot (l + \lg n)$.
 $\Rightarrow p = O(ln)$ exists!

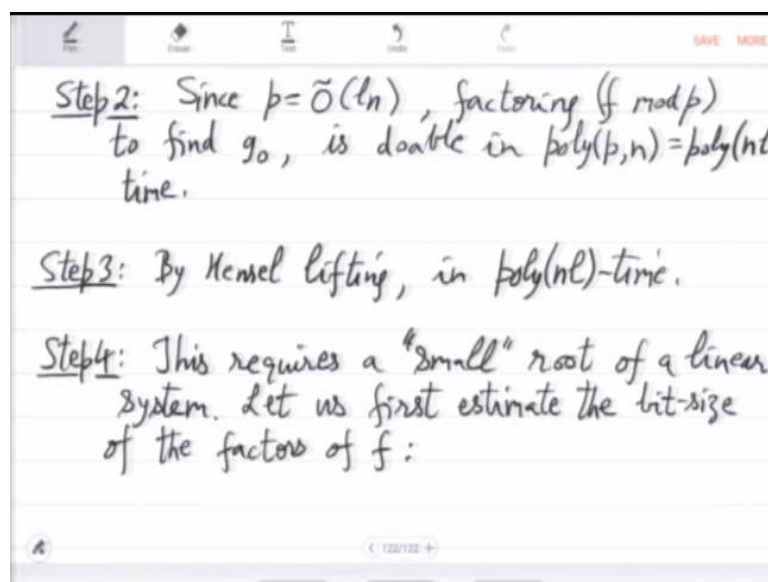
So in step 1 what is the prime that will work, right? So we want to look at the case when f given input polynomial f is square-full. So suppose it is square-full, then what you can do is you can compute the GCD. Just compute the integral GCD of f and f' and that will for sure factor f , right. So this is an easy case. Now it may happen that f is square-free but mod p it is square-full.

So if it is square-full mod p then the GCD of f and f prime mod p will be non-trivial which implies that the resultant of f and f prime which is now a number, right when you take resultant with respect to x , so this is x . When you eliminate x you are left with an integer and that integer will be divisible by p , it is $0 \bmod p$. So that is r . So what you want is you want a prime p such that p should not divide the resultant r and p should not divide a_n the leading coefficient, okay.

So we want a prime number p that does not divide a_n times r . Note that a_n times r is nonzero to begin with; a_n is obviously nonzero because the degree was n of f ; r is nonzero because f is assumed to be square-free. So the resultant will be nonzero integer. So just pick a prime which does not divide it, okay. So a_n times r is by the definition of resultant it is as big as this.

And you can see that this bounds the number of primes by log of this bound. So definitely a prime around 1 times n will exist and can be found, okay. So you have work with that prime. That is the analysis of step 1. Okay, now let us move forward to step 2. Okay, so what happens in step 2?

(Refer Slide Time: 06:54)



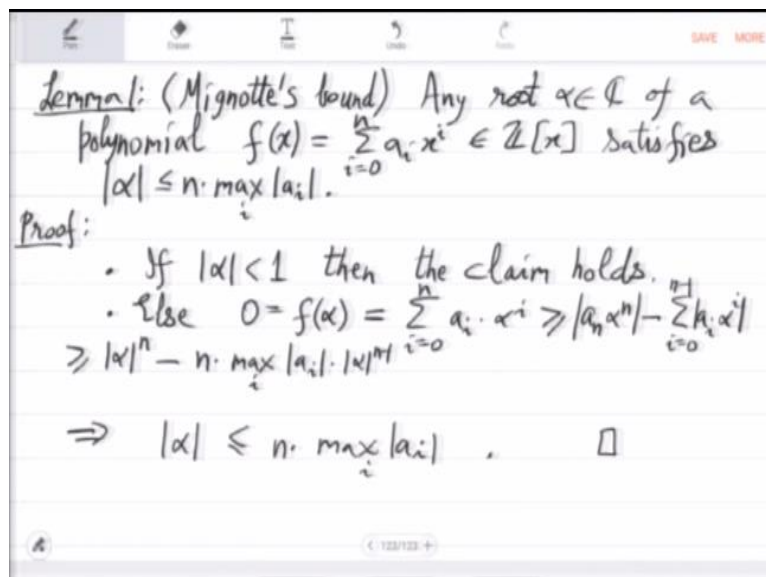
So in step 2, you are just factoring $f \bmod p$. So since p is equal to $\tilde{O}(1/n)$ it is around 1 times n in magnitude. Factoring $f \bmod p$ by Berlekamp factoring $f \bmod p$ to find g_0 is doable in polynomial in prime p and the degree of f , right which is overall polynomial in n^l . So in this much time, which is truly polynomial time in the input size you are able to find g_0 by Berlekamp algorithm, right.

So recall Berlekamp for this. So that completes step 2. Step 3 is Hensel lifting. So let us analyze that. Again recall the k times Hensel lifting. So that we have seen is a polynomial, deterministic polynomial time algorithm. It is polynomial in k and the degrees and the bit sizes, right. So this is by Hensel lifting lemma. This is doable again in polynomial in $n+1$ time, right because k is also picked to be quite small.

So everything can be done in polynomial in $n+1$ time. Let us move to step 4. So in step 4 is this linear system solving. But there is a catch now. So indeed in terms of the unknowns of \tilde{g} and l_k this is a linear system. But not any solution of the linear system will work. So you want a solution which is small in bit size, right. So you cannot just invoke a linear system solver. So this is actually the hard part.

This is what will lead us to something new. So this requires a small root of a linear system. So we will see, in the next lectures we will see how to find a small root of a given linear system. But let us first see why a small root will exist, right? It may not exist. So let us first see the existence proof, then we will move on to the constructivity. So let us first estimate the bit size of the factors of f . So we will prove the following lemma.

(Refer Slide Time: 11:24)



Lemma: (Mignotte's bound) Any root $\alpha \in \mathbb{C}$ of a polynomial $f(x) = \sum_{i=0}^n a_i x^i \in \mathbb{Z}[x]$ satisfies $|\alpha| \leq n \cdot \max_i |a_i|$.

Proof:

- If $|\alpha| < 1$ then the claim holds.
- Else $0 = f(\alpha) = \sum_{i=0}^n a_i \cdot \alpha^i \geq |a_n \alpha^n| - \sum_{i=0}^{n-1} |a_i \alpha^i|$

$$\geq |\alpha|^n - n \cdot \max_i |a_i| \cdot |\alpha|^{n-1}$$

$$\Rightarrow |\alpha| \leq n \cdot \max_i |a_i| \quad \square$$

This is called Mignotte's bound. Any root α , any complex root α of a polynomial $f(x)$, so $a_i x^i$ to the, a_i are the coefficients and there are $n+1$ of them, degree is n , right and these are all integers. So any root, so you know that over

complex polynomial f will completely split into complex roots. The question that Mignotte's bound answers is how big is the magnitude of each of these roots.

So it says that the magnitude is, the magnitude is at most n times the max coefficient, okay. So ultimately the statement is very natural. It is saying that any complex root you pick of a polynomial, the magnitude is bounded by the magnitude of the coefficients times n , which is the degree. And the proof is equally simple. So what you, you basically divide into two cases.

So if the if you are looking at a root α whose magnitude is less than 1, right then of course, so in that case what do you do? Well, in this case the bound n times max bit size that is at least 1 right. So this is trivially true. Then the bound holds trivially. Otherwise, so otherwise the norm of α is greater than equal to 1. So in that case, let us look at the evaluation of f at α .

Since α is a root, obviously it is zero. But on the other hand, you also know that $\sum a_i \alpha^i$ is at least. So since the norm of α is at least 1 this sum the bigger chunk of this sum comes from α^n , okay. So let us isolate that. $a_n \alpha^n$ minus the rest, okay. So this is the this is the lower bound on this sum $\sum a_i \alpha^i$.

We have isolated the highest term and the from the rest looking at the difference of that, which is so which is greater than equal to α^n because a_n will be at least 1, right. It is an integer. So the magnitude is at least 1, it is not 0. So α^n minus $\sum_{i=0}^{n-1} a_i \alpha^i$ is at most so actually we can remove the sum, use the max instead. This is at most n times max magnitude over all the i 's times α^{n-1} , right.

So here we are doing two things. One is that the norm of a_i will be at least, will be at most max over all the a_i 's. And α^i will be norm of that will be at most α^{n-1} . This is because α 's norm is at least 1, right. So this and then there is a minus sign. So the inequality is correct. And remember 0 is greater than equal to this, right.

So which means that what you get is alpha is less than equal to n times max of a i, right which is what you wanted to show. So this is the so it is a simple proof. Just by just look at the evaluation of f at alpha and you will get this, okay. So this means that the roots are small. Now from this you can talk about the factors, factors will be small too.

(Refer Slide Time: 17:55)

Lemma 2: Any factor g of f has coefficients of magnitude at most $2^{(l+ln-1)n}$.

Proof:

- Let $g(x) = \prod_{i=1}^m (x - \alpha_i)$, $\alpha_i \in \mathbb{C}$
- $\text{Coeff}(x^{m-j})(g) = \sum_{S \in \binom{[m]}{j}} \prod_{i \in S} (-\alpha_i)$
- with magnitude $\leq \sum_S \prod_{i \in S} |\alpha_i|$

[Lemma 1]
 $< \binom{m}{j} \cdot (n2^{l-1})^j < (1+n2^{l-1})^m < 2^{(l+ln-1)n}$. \square

So that is lemma 2. Any factor g of f has coefficients of magnitude at most 2 raised to 1 plus log n minus 1 times n . Okay, so what this is saying is that what did you get in the in Mignotte's bound? So you got that the bit size I mean okay, the magnitude of a root, complex root is around the bit size of the coefficients, right. So it is kind of n times 1. It is kind of n times 2 raised to 1.

For the factor you are getting something slightly larger, okay. This is 2 raised to 1 and then another power of n . It is 2 raised to 1 times n , okay. It is slightly more but remember that this is the magnitude. So the bit size is only $1 + n$, right. So in terms of the input size, the bit size is pretty small. So factors cannot be too large. That is the bottom line of this statement, okay.

So the way you will show it is by looking at the complex roots of the factors and applying lemma 1. So factorize g into roots. Let us say there are m roots, degree is m . α is a complex numbers but g is an integral factor. So these complex roots they multiply together and then you get integers. And for these integers we will have a bit size upper bound. So consider this g .

Look at the coefficient of coefficient of x to the m minus j in g is what? So this is like picking j of these alphas and summing up. So basically pick j of these or subset s of size j out of 1 to m and then multiply these alphas, okay. So this is the coefficient of x to the m minus j . And using lemma one you can bound this with magnitude less than equal to sum over all these subsets then product norm of α_i , right.

This is by lemma 1, which is smaller than number of subsets is m choose j times look at a product now, product α_i . So lemma 1 says that so product α_i you invoke lemma 1 and you will get n times 2 raised to $1 - 1$ raised to j , right. So this is true for every j and overall each of these things they will be smaller than this. j goes from 0 to m . So that is the upper bound, right. This is by the binomial expansion.

So m choose j times n 2 raised to $1 - 1$ raised to j is at most this expression which itself is smaller than 2 raised to 1 plus $\log n$ minus 1 and raised to m , m is at most $n - 1$. So we can safely put n here, okay. So that is the bound for the coefficients of arbitrary factors of a polynomial, okay.

So ultimately what we have is once you bound the bit sizes of the coefficients and the degree, so the input polynomial once you know its bit representation size its bit size, you also know an upper bound on the bit size of the factors, okay. You have a decent upper bound on that as well. Okay, so that finishes the analysis for step 4, right. So step 4 we know that if there is a factor of f , integral factor of f , it will have bit size smaller than n times 1 plus $\log n$ right.

This is what we have shown and step 4 is trying to find that. Okay, so if f has a factor then g_k will lead you to that, that factor $g_{\tilde{k}}$. So existence is now clear. How do you construct this that is not clear. Okay, let us now move to step 5. Why is step 5, what is step 5 doing?

(Refer Slide Time: 25:16)

Step 5: • If \tilde{g} exists in Step 4, and $(f, \tilde{g}) = 1$,
 $\exists u, v \in \mathbb{Z}[x] : u \cdot f + v \cdot \tilde{g} = \text{res}(f, \tilde{g}) \neq 0$
 $\Rightarrow u \cdot g_k \cdot h_k + v \cdot g_k \cdot l_k \equiv \text{res}(f, \tilde{g}) \pmod{p^{2k}}$
 $\Rightarrow g_k \cdot (u h_k + v l_k) \equiv \text{res}(f, \tilde{g}) \pmod{p^{2k}} \quad \text{--- (i)}$
 • Note: $|\text{res}(f, \tilde{g})| < (2n+1)! \cdot (2^{l-1})^{nH} \cdot (2^{(l+g_n)n})^n$
 $\ll 2^{2nl} < p^{2k}$
 \Rightarrow RHS in eqn (i) is a nonzero constant,
 while LHS is a multiple of $g_k(x)$.
 \Rightarrow The contradiction implies that Step-5
 factors f , if \tilde{g} exists. \square

So let us go there. So this step also needs a detailed analysis. It is a very important step, but you have seen this before in bivariate factorization. So the analysis will be very similar, actually it will be resultant based. So if \tilde{g} exists in the previous step and let us say for the sake of contradiction that it does not factorize f , okay. So step 4 found a \tilde{g} with appropriately small coefficients.

But in step 5 the GCD of f and \tilde{g} is 1. So what we want is actually a contradiction. So let us see what how the contradiction will appear. So they being coprime means that there exist by Euclidean GCD algorithm extended Euclid GCD. They exist u and v such that $u f + v \tilde{g}$ is equal to, so notice that I am asking for u, v to be integral polynomials, right.

So if they were rational polynomials then $u f + v \tilde{g}$ would have been 1. But since they are integral it will be the resultant, which is nonzero, okay. So since f and \tilde{g} you are assuming to be coprime the Bezout identity will look like this, $u f + v \tilde{g}$ equal to this number, resultant now is a number and it is nonzero. Okay what do you do here? So as you can guess, you have information about how f factors and how \tilde{g} factors.

So use that information. So you will get u times f is $g_k h_k$ and \tilde{g} is $g_k l_k$, which is obviously resultant modulo p raised to 2 raised to k , okay. So modulo prime power you know how f and \tilde{g} factor and this gives you this nice congruence where g_k is outside. So you get this, okay. So RHS I have not changed. LHS we see

that $g \mid k$ divides. Now this is absurd because on the RHS, you have a number and on the LHS you have a polynomial, which is not a number, right?

It is non constant because $g \mid k$ is non constant. So the only way this can happen is if both LHS and RHS are zero, right? So can RHS be 0? Can this number be zero mod prime power? That is the question. So let us analyze that. When is it 0? What does it mean? So note that the resultant it was what is the magnitude of this resultant number? So degree of f is n . Degree of \tilde{g} is $n - 1$.

So number of coefficients are $n + 1$ and n . So it is $2n + 1$ dimensional matrix when you want to compute the resultant. And how big are the entries? So for f the entries are smaller than 2^{l-1} . For \tilde{g} the entries are by construction. It is 1 plus $\log n$ times n and then the whole thing n , right. That is what upper bounds the resultant number which you can see is smaller than 2 raised to how much do you get?

So this comes out to be around n^2 times 1 , right and we have picked 2 raised to k sufficiently large. So let me say this is smaller than 2^{2n^3} which is smaller than p^{2^k} , right. This is why we picked 2 raised to k to be large enough or k to be large enough. So this prime power is actually much larger than the resultant. So there is no scope of non-zero resultant becoming zero, right.

So in equation one, RHS is nonzero. So this implies that RHS in equation 1 is a nonzero constant in fact, while LHS in equation 1 is a non-constant polynomial or is a multiple of $g \mid k$, $g \mid k \mid x$. Okay so RHS is a nonzero constant. LHS is a multiple of a polynomial $g \mid k \mid x$. Now multiples of polynomials are again non constant polynomials except the zero multiple. But it is not a zero multiple.

It cannot be the zero multiple because RHS is nonzero. So that is the contradiction between LHS and RHS. Okay, so that gives you the contradiction. And this contradiction means that, the contradiction implies that step 5 factors. Step 5 factors f if \tilde{g} exists. Okay, so whenever step 4 succeeds, step 5 also succeeds. Okay, so this is an amazing property.

So all we have to do is work out step 4. Once we have worked out step 4, step 5 will automatically succeed and factorize. Okay, so we have covered all the steps except obviously step 4 details we have to now work out. Okay, so step 1, step 2, step 3, 4, and 5 is what we have completed. So now let us focus on this missing part which is step 4. The algorithm hidden in step 4, right? How do you actually compute step 4.

(Refer Slide Time: 34:49)

How do we compute \tilde{g} (with "small" coeffs)?

— let g_k be of $\deg = n' < n$. Unknown polynomials are $\tilde{g} = \sum_{i=0}^{n-1} c_i \cdot x^i$ & $l_k = \sum_{i=0}^{n-1-n'} \alpha_i \cdot x^i$ s.t.

$$\tilde{g} \equiv g_k \cdot l_k \pmod{p^{2^k}}$$

$$\Rightarrow \sum_{i=0}^{n-1} c_i \cdot x^i = \sum_{i=0}^{n-1-n'} \alpha_i \cdot (x^i g_k) + \sum_{i=0}^{n-1} \beta_i \cdot (p^{2^k} x^i) \quad (ii)$$

△ Find integral $\bar{c}, \bar{\alpha}, \bar{\beta}$'s in eqn(ii) s.t. $\|\bar{c}\| = \sqrt{\sum c_i^2}$ is "small" $< 2^{(l+qn) \cdot n}$

With small coefficients, that is the thing. So there is a linear system. We can find a root of the linear system. But then how do you find a small root? That is the new thing you have to do. So let us set up some notation to understand this. So let g_k be of degree n prime, obviously between 1 and $n - 1$. Now the unknown polynomials are \tilde{g} which is let us say $c_i x^i$ to the i . And l_k . Let us give it the coefficients α_i .

And \tilde{g} is g_k times l_k , right. So \tilde{g} has degree $n - 1$ at most. So l_k will have degree $n - 1$ minus n prime, fine. So you have to find these with the constraint satisfying the constraint \tilde{g} is congruent to g_k times l_k mod p raised to 2 raised to k . And the unknown coefficients are c_i α_i . And in this equation it is $\tilde{g} l_k$, right. So g_k is completely known. You want to find the coefficients of $\tilde{g} l_k$.

But then the coefficient of there is a condition on the c_i 's that it should be small, okay. Let us do some more jugglery. So let us rewrite it as $\sum c_i x^i \cdot l_k = \alpha_i x^i g_k$. That is the RHS I am writing, right. So this is what it is, but I want to remove the modulus. For some reason I want to make it exact. So then I have to introduce multiples of prime power. So let us do that.

So p raised to 2 raised to $k \times$ raised to i okay. So let us call this equation 2 okay. So here the unknowns are, the unknowns are c_i , α_i , and β_i . And it is clearly they clearly are roots of a linear system and they are also integers. That is what we are looking for. So find integral \bar{c} $\bar{\alpha}$ $\bar{\beta}$ in equation 2 such that \bar{c} is small. The vector \bar{c} is small. So the norm of the vector \bar{c} is $\sqrt{\sum c_i^2}$, square root right.

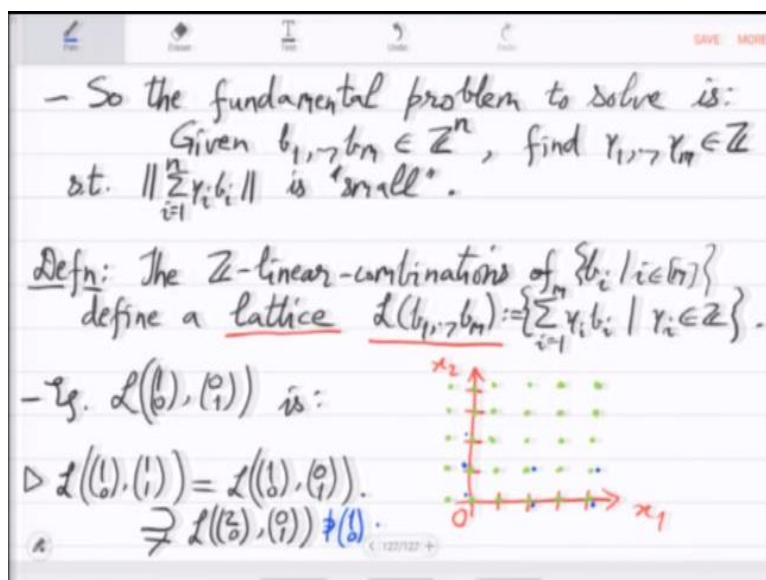
That is the Euclidean norm. So I want this to be small. How small? Well, so step 4 said it should be smaller than $2^{1 + \log n}$ times n . In step 4 what did we have? The coefficients have bit size $2^{n(1 + \log n)}$, okay. So now I am replacing it by a slightly better condition because I know that such a \bar{c} exists, right. By the previous, if you look at the previous proof, in this step 5 existence proof, sorry not step 5, step 4.

In the step 4, if you look at lemma 2, the integral factors coefficients were bit size $1 + \log n$ times n right. So each of the c_i is this much. And if you look at the norm of the \bar{c} vector, it is definitely smaller than $2^{1 + \log n}$ times n . So this is blue bound okay. So that is the question. So we have reformulated the question.

You are given these, you are given a linear system with the unknown variables \bar{c} , $\bar{\alpha}$, $\bar{\beta}$ and you want an integral solution such that the norm of \bar{c} is smaller than this, $2^{1 + \log n}$ times n . So let us now jump into this linear system small root finding question directly and forget about the application that we saw before, okay.

Let us just look at this question and we will invest the next this class and the next class to answer this fundamental question. We will use something that is conceptually new. You probably have not seen those things.

(Refer Slide Time: 43:09)



So the fundamental problem is you are given vectors with the, you are given m vectors, let us say the ambient space is n . So you are given m vectors. Each vector has n coordinates. The coordinates are all integral. Okay, this is what you are given. Now okay let me set up the correspondence. So b_1 to b_m will basically correspond, in this equation 2 it will correspond to the basis vectors of the solution set, okay.

So this equation 2 is a linear system. So you can solve it and you can find a basis of the roots. Those basis vectors I am now calling them b_1 to b_m , okay. So that you can compute, you have it. But that is not what the answer that you want. So you actually want their combination. You want their integral combination such that the norm is small.

Okay, so the fundamental problem to solve is that given m vectors in the n ambient space can you find a small vector that is spanned by them using integral coefficients, okay. So this γ_1 to γ_m being integers is important okay. So basically to get from this equation to that fundamental problem you find a basis, integral basis of this and then just ask the question for that basis.

The $\sum \gamma_i b_i$ that you will find whose norm is small will correspond to \bar{c} . Okay, so you are interested in finding that \bar{c} . So these \mathbb{Z} , so we call $\sum \gamma_i b_i$, we call this a \mathbb{Z} linear combination, integral linear combination and the set of these combinations is called a lattice. So the \mathbb{Z} linear combinations of b_i define a lattice, okay.

So just like q linear combinations, rational linear combinations will define a vector space. \mathbb{Z} linear combinations define a lattice. And we will use the following notation. So this $L = \mathbb{Z}b_1 + \mathbb{Z}b_2 + \dots + \mathbb{Z}b_m$ is the lattice spanned by these vectors such that γ_i 's are integral. Okay, so for example, how does the lattice look like pictorially, right. So a vector space you know for example, one dimensional vector space will just be the real line right and two dimensional vector space will be the real plane.

So in the same spirit what are the one what is the picture for one dimensional two dimensional lattices? So if you consider $(1, 0)$ and $(0, 1)$ these elementary vectors right the x and y kind of axis. So this will be so this is $1, 2, 3, 4$. Okay, so this is the let us say x_1 axis, x_2 axis the origin. So where will the lattice points be? So of course, $(1, 0)$ is a point and $(0, 1)$ is a point and in their integral combination origin is also there.

So that is also a point but then you can double $(1, 0)$ right. So this is a point, you can triple it, you can make it four times and similarly here in the x_2 . Now what are the other lattice points other than the axis? Well, so this is also a lattice point right $(1, 1)$ because you can sum up and by summing up other things you will get these other points, right. So that is it. So this these green points these are exactly the lattice points.

And they are in all the quadrants, okay. So because you can also, you can not only add you can also subtract. All integer combinations are allowed, so it is on all the sides. So you can also go here, right. So this is how a lattice looks like. As you can see the, as you can deduce from the picture it is a very discrete object. So you really have to jump from one point to the next. There is a big gap between them.

And this is happening because you are taking integral combinations and not real combinations, okay. These are very useful objects as you will find out in the next lectures. Okay, what is the lattice spanned by $(1, 0)$ and $(1, 1)$? What do you think, will it be different? It would not be different because $(1, 1)$ is just $(1, 0)$ plus $(0, 1)$, right. So you can again add and subtract and you will get the previous lattice, okay.

On the other hand, if you look at $(2, 0)$, $(0, 1)$ what do you think about this lattice? Do you think this is the same as before? No, it is not the same as before because you have

actually doubled one basis vector and that makes this lattice coarser, okay. So in the set notation this is a proper this is a sub lattice, okay. So if you scale up a basis then you will get actually a lattice which is coarser, okay.

There is for example, there is no way to get 1 0 in this, right 1 0 is absent. It does not contain 1 0. So that but everything that this contains the previous lattice also contains. So what you will get is actually a coarser lattice like this. On the x 2 side you will get everything, okay. So what is the question in this lattice that we are interested in? So the question is given a lattice find a vector in the lattice which is small, right.

So for example, you can ask the idealized question find the smallest vector in the given lattice. So if you look at this green lattice, the smallest vector here is 1 0 or 0 1 right. Its multiples cannot be the smallest but 1 0, 0 1 or minus of that, those have length 1 and anything else will have length more than 1, right. So here pictorially it was easy to see, but as m or n increase then it becomes a challenging question, okay.

So you basically have to find the right integral combination so that the length is minimized. So that is called the shortest vector problem.

(Refer Slide Time: 54:48)

- Shortest vector problem (SVP): Find a vector $\vec{v} \in L(b_1, \dots, b_n)$ st. $\|\vec{v}\| = \min_{0 \neq \vec{u} \in L} \|\vec{u}\|$.

▷ [Ajtai '98] SVP is NP-hard.
[Micciancio '98] constant-approx. of SVP is NP-hard.

- But, we need merely a 2^n -approximation for Step 4!

→ We'll develop this approximation algorithm (L^3 = Lenstra-Lenstra-Lovász)

So shortest vector problem is SVP. Find a vector v in the lattice such that the norm of v is minimized. So let me write this properly such that the norm of v is equal to the minimum over all the norms. So for all the vectors available in the lattice the smallest

norm obviously excluding 0. 0 is always there. So the positive norm, smallest positive norm. That is the SVP question.

This is actually, it may not be immediate, but this is a very hard question and not very long ago Ajtai was the first one to show that this is NP-hard. And soon after Micciancio showed that constant approximability of SVP constant basically in particular he showed that if you want a vector that is at most square root 2 times the shortest vector, so an approximately shortest vector that is also NP-hard.

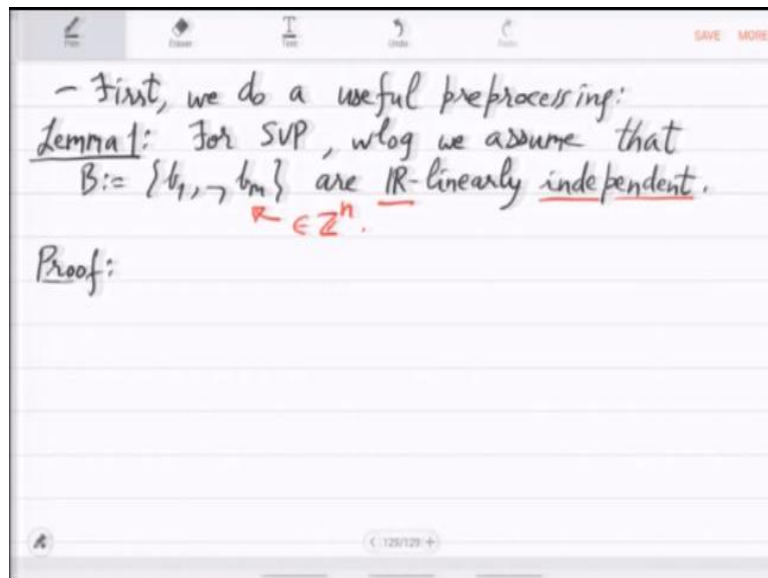
Okay, so this settles the inapproximability of SVP. Not only is it NP-hard to compute exactly, even you cannot even approximate it, right. So then it would seem that the step 4 where we want to find a short vector satisfying a linear system, that question may be NP-hard. It may we may have reduced to a hard question. Well, fortunately that is not the case because we need merely a 2 raised to n approximation for step 4. Why is that?

Well, you look at the look at the ambient space. So that ambient space was in our case it was how many c bars are there? Right that is only n and for that ambient space of n the step 4 requires a c bar to be around 2 raised to $1/n$, right. And 2 raised to $1/n$ is also it is an upper bound for the coefficients in the input, right. So in terms of n actually, it is an exponential approximation that you are looking for.

So there is a, the demand on c bar is very relaxed. And that will save us. Okay, yeah, so we will basically see this, we will develop this approximation algorithm. Okay, and this was first by L cube. So Lenstra – Lenstra – Lovasz. So this is the main content of L cube algorithm that they give an approximation algorithm to solve the SVP problem. And then as an application as we have seen step 4 they can factor integral polynomials in deterministic polynomial time, okay.

So this is what will develop in the end. We will first start with a preprocessing step.

(Refer Slide Time: 1:00:46)



So first we do a useful preprocessing. So lemma 1, the first lemma that we will show is without loss of generality. We assume that the given vectors right which were b_1 to b_m . Till now we have not said anything about them except that they lie in this ambient space \mathbb{Z}^n . So we can actually assume and it will not change anything. So we can assume without loss of generality that they are linearly independent, okay.

So we can assume them to be linearly independent even over reals, the real field, okay. So remember that in the input b_1 to b_m could have been dependent in some complicated way, right. You could have b_1, b_2 and then you may have $2b_1$ plus $3b_3$ right or you may have $2b_1$ plus $3b_3$ and then you may have $10b_1$ plus $100b_2$ and so on. So it is not really clear whether using integral combinations of that you can get back to b_1, b_2 .

But in this lemma we will show that if there is a dependence amongst b_1 to b_m , then you can actually reduce the input instance from m to $m - 1$. And if again there is a linear dependence you can reduce it to $m - 2$ and so on, okay. So you can work with a smaller instance. One important assumption here will be that these are integral. This will be important in the proof that we start with the integral vectors.