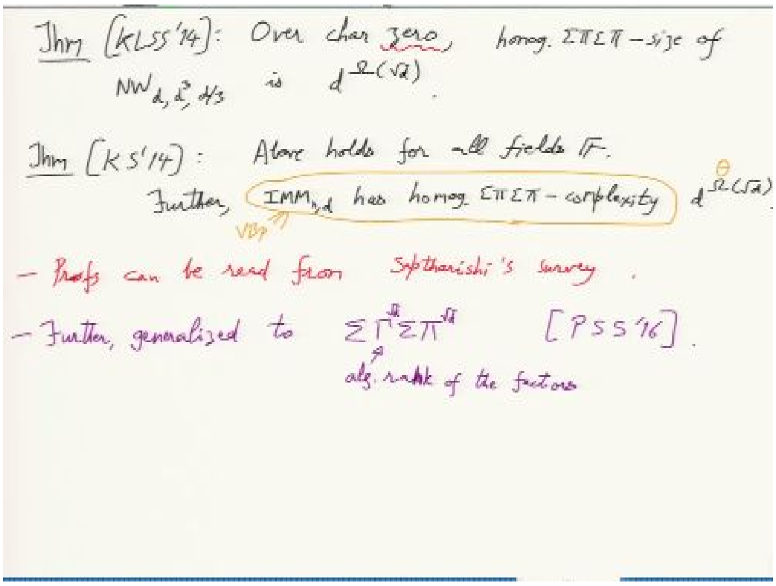
Arithmetic Circuit Complexity

Prof. Nitin Saxena
Department of Computer Science and Engineering
Indian Institute of Technology-Kanpur

Lecture - 22

(Refer Slide Time: 00:16)



Thm [KLSS'14]: Over char zero, homog. $\Sigma\Pi\Sigma\Pi$ - size of $NW_{d,d^3,d/3}$ is $d^{\Omega(\sqrt{d})}$.

Thm [KS'14]: Above holds for all fields $\mathbb{F}$.
Further, $\mathrm{IMM}_{n,d}$ has homog. $\Sigma\Pi\Sigma\Pi$ - complexity $d^{\overset{\Theta}{\Omega}(\sqrt{d})}$.
VBP

- Proofs can be read from Saptharishi's survey.
- Further, generalized to $\Sigma\Gamma^{\sqrt{d}}\Sigma\Pi^{\sqrt{d}}$ [PSS'16].
alg. rank of the factors

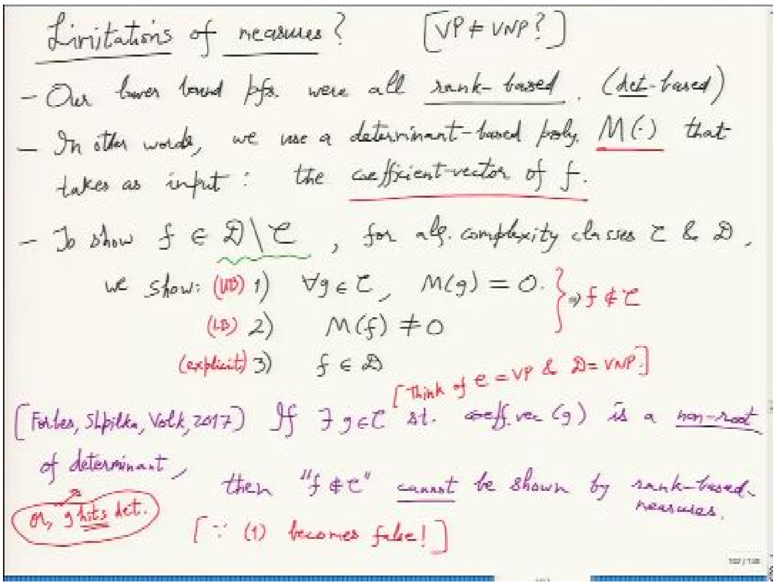
So in the last class we saw, we did not go into the second part of the details, but we defined the measure and we saw an upper bound and if you complete the lower bound then you will get that this simple polynomial, which is VBP polynomial, right iterated matrix multiplication. This has homogeneous depth-4 complexity optimal. So this is $d^{\Theta(\sqrt{d})}$.

Because you can implement it in this much size and we have shown that nothing better can be done. And this method was generalized slightly to other models and also for other fields. So this was further generalized to some model called $\Sigma\Gamma^{\sqrt{d}}\Sigma\Pi^{\sqrt{d}}$. So this Γ is basically referring to again product of sparse polynomials.

But their algebraic rank is bounded by $\sqrt{d}$. So instead of their number, so their number may be anything maybe arbitrary, but their algebraic rank is small. So this actually refers to the algebraic rank of the factors. $\Gamma^{\sqrt{d}}$ refers to algebraic rank of the factors being $\sqrt{d}$. It does not talk about fanin, it talks about the algebraic rank.

So this was done in a paper by Pandey Sinhababu and myself 16. So this is more algebraic. So yeah, I mean this measure has been extended in several ways, the measure of shifted partials. Any questions? Okay. So now what we will do is just quickly go through an interesting point that was made by some authors in 2017, which refers to the limitation, potential limitation of these techniques.

(Refer Slide Time: 03:02)



Limitations of measures? [VP ≠ VNP?]

- Our lower bound pfs. were all rank-based. (det-based)
- In other words, we use a determinant-based poly. $M(\cdot)$ that takes as input: the coefficient-vector of f.
- To show $f \in \mathcal{D} \setminus \mathcal{C}$, for alg. complexity classes $\mathcal{C}$ & $\mathcal{D}$,

we show: (UB) 1) $\forall g \in \mathcal{C}$, $M(g) = 0$.
(LB) 2) $M(f) \neq 0$
$\} \Rightarrow f \notin \mathcal{C}$
(explicit) 3) $f \in \mathcal{D}$

[Think of $\mathcal{C} = VP$ & $\mathcal{D} = VNP$]

[Forbes, Shpilka, Volk, 2017] If $\exists g \in \mathcal{C}$ s.t. coeff.vec(g) is a non-root of determinant, then "$f \notin \mathcal{C}$" cannot be shown by rank-based measures.

Or, g hits det.

[$\because$ (1) becomes false!]

So the question is whether you can come up with a creative measure. So first come up with a vector space and then look at the rank of that vector space and using this strategy can you show VP different from VNP, right. So this is the ultimate goal. So can this be done by just being more creative about the vector space and about the measure?

So since it will be a vector space based measure, it is fair to say that we are looking at determinant, right. Because rank of vector space is essentially determinant of some vectors. So our lower bound proof they were all ranked based. So which is also determinant based and to be, in particular, what we were doing was the following. So we use our determinant based polynomial M that takes as input the coefficient vector of f.

So the determinant will be of what? It will be applied on the coefficient vector of the polynomial which you want to study, right because that is all you can compute determinant of. So the vector space basically, for example, if you look at the space of partial derivatives right so that is also exactly of this type that it is basically playing with the coefficients; with the coefficient vector of f.

And then it extracts other vectors from this and then it looks at the determinant of all these vectors which have been extracted, right? So it is basically we can say that this determinant based polynomial M is acting on the coefficient vector of f. Yes. Minor. Rank is obtained by minor, which is also a determinant. So to show that $f \in D \setminus C$.

This is what we want to show when we are interested in explicit in a polynomial that is both explicit and hard. So it is explicit for D and it is hard for C, right. So it is in $D \setminus C$. This is what we want to show. That is the target in all these lower bound proofs. So to show this for algebraic complexity classes C and D, what is done? So what is done is three things. So first we show that for any polynomial in C, M(g)= 0.

So this is the upper bound part of our strategy. So when we show that this $\Sigma\Pi^{\sqrt{d}}\Sigma\Pi^{\sqrt{d}}$ shifted partials on this is small, that when we show that upper bound, we are basically looking at some determinant M. And we are saying that all these polynomials g that can be computed by these models M vanishes. So that is the upper bound. That is the first thing.

Second is you show that f violates this. That is the lower bound part. So f you have to then design creatively, so we looked at determinant for example, or iterated matrix multiplication polynomials, that is f. So once we have shown 1 and 2, what do you get? This means that f is not in C, right. Clearly this implies that f is not in C. And third is which basically follows in the previous proofs by construction that f is in D.

That is the explicitness. So we want these three properties and then we have an explicit lower bound, right. So the observation made by Forbes Shpilka, and Volk in

17. So their observation was that if your complexity class C is complicated enough, right in the worst case think of VP. So then VP we know has pretty complicated polynomials. Even determinant is there and many other things are there whose coefficient vectors are supposed to be pretty complicated.

So if you believe that there are coefficient vectors, on which a big determinant will not vanish or in other words they are coefficient vectors that will be non-roots of determinant right then property 1 can never be satisfied. It cannot be true that for every g in the in C, M(g) vanishes. If you believe that your coefficient vector of g for some g in VP is hard enough or is complicated enough then determinant will not be able to kill it.

So property 1 cannot be satisfied. So it is a conditional thing. If there exists a g such that the coefficient vector of g is a non-root of determinant. If this is assumed it is a conjecture, it is a hypothesis. So if you assume this hypothesis, then “f not in C” cannot be shown by rank based measures. Because, simply because property 1 cannot is false. So since property if you cannot you would not be able to show even property 1 if C is a complicated class.

Hence this whole strategy you would not be able to make it work. Is that clear? So assuming that there are these polynomials in VP whose coefficient vectors are complicated, by which we mean determinant does not vanish right then these measures will always be limited, you cannot separate VP from VNP. So the C here is think of it as; think of C as VP, and D as VNP.

This is the motivating example. But then VP is not really a simple complexity class. It is a class that has all types of difficult looking polynomials mean they happen to be in VP, but those proofs are always non trivial. And the polynomials have exponentially many monomials. It is unlikely for determinant, if M is determinant.

So whatever determinant based polynomial you will design as M it is expected that there will be a g in VP whose coefficient vector will keep it nonzero. So that is also called - there is a g that hits determinant. So g will hit determinant. I mean determinant is a nonzero polynomial and if you look at the coefficients that appear in g and use them to evaluate determinant, determinant will be nonzero.

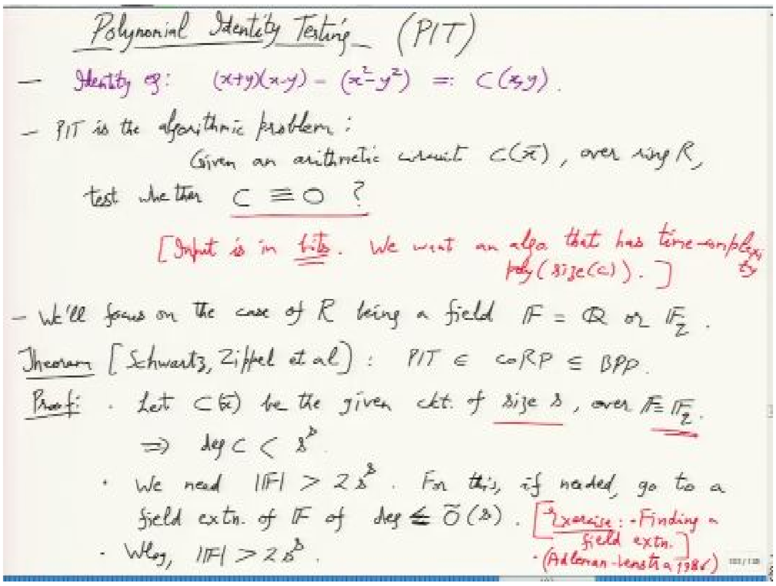
I mean that is expected because the, because g has all sorts of coefficients. There is no reason why determinant will vanish at this point .Yeah, so this is the core of their paper. But the paper has many other things and it formalizes this and also implements many known hitting sets in their formulation. So that will be covered by Abibhav at a later date. Any questions? But this is the core.

This is the reason why people may believe that these measures will not generalize. So what do you think will generalize? If these measures do not generalize then does it mean you can never prove VP different from VNP? Yeah, that is where I will go next. But here one loophole that you can employ is I mean in the measure do not keep your measure to be strictly a polynomial of the coefficient vector.

So for example, you can make it a non-polynomial by saying that I will also go into the bits of the coefficients. So whether the ith bit of the jth coefficient is 1. So if you ask these bit, if you force bit operations or these checks with bit operations, that will not be a polynomial, that will not be a determinant polynomial because it is going it is looking at not just the variable x_1 but actually when you substitute the coefficient in x_1 it is going inside the bits or the string of x_1.

So then these arguments will not hold anymore, right. So you might talk about inequalities whether the coefficient is greater than zero or whether a bit is 1 or 0. So those things are not simply a question of determinant. Yeah, so all sorts of things you can do. That is not covered in these limitations. But we were not doing that in the previous proofs. As Prateek wants that brings us to polynomial identity testing.

(Refer Slide Time: 17:22)



Polynomial Identity Testing (PIT)

- Identity eg: $(x+y)(x-y) - (x^2-y^2) =: C(x,y)$.
- PIT is the algorithmic problem:
  Given an arithmetic circuit $C(\bar{x})$, over ring R, test whether $C \equiv 0$?
  [Input is in bits. We want an algo that has time-complexity poly(size(C)).]
- We'll focus on the case of R being a field $\mathbb{F} = \mathbb{Q}$ or $\mathbb{F}_2$.

Theorem [Schwartz, Zippel et al]: PIT $\in$ coRP $\subseteq$ BPP.

Proof: · Let $C(\bar{x})$ be the given ckt. of size s, over $\mathbb{F} = \mathbb{F}_2$.
$\Rightarrow \deg C < s^s$.
· We need $|\mathbb{F}| > 2s^s$. For this, if needed, go to a field extn. of $\mathbb{F}$ of $\deg \leq \tilde{O}(s)$. [Exercise: - Finding a field extn.] · (Adleman-Lenstra 1986)
· Wlog, $|\mathbb{F}| > 2s^s$.

"Professor - student conversation starts" But is there any approach for either related to codes because, sure whether they are r approaches, I do not know. But you can try, prove a new lower bound. I thought if there is some. No, not currently but it is not ruled out. Feel welcome. **"Professor - student conversation ends".**

So PIT is the question of given a circuit you have to say yes or no depending on whether the circuit is identically 0, right. So the word identity here just means whether your circuit is encoding an identity, right. So that is what this word identity means. So identity example is $(x+y)(x-y)-(x^2-y^2)$. So this may be your circuit and this is identically 0.

Although if you look at the intermediate computations, nothing is zero. But it is at the very end that you get, at the very end of the formula, you will see that everything will cancel out. So that is the challenge that it is about representation. It is not about the polynomial that is because the polynomial is just zero. So which is why this question is not covered under standard algebra or standard algebraic geometry.

Because it is really about the representation. So in this case for example, this is a formula. So I have given you a formula where everything is nonzero, but in the end of the calculation you see zero, monomials cancel out, right. So it is a question inherently about the representation. So PIT is the algorithmic problem. Given an algebraic circuit. Let me not say algebraic let me say given an arithmetic circuit $C(\bar{x})$ over a ring.

So coefficients come from a ring R of constants. Test whether C is identically 0, okay that is the question. It is a decision problem. When we say given a circuit, arithmetic circuit we mean that you can see all the gates and you can see wires and you can see the constants on the wires. As an algorithmic problem it will make sense to also count the bit representation right.

So when there is a constant it should be represented in bits. That is one thing we can assume here that input is in bits. In an algebraic, so in algebraic complexity, we do not worry about the constants, constants can be anything. But when you ask for an algorithmic problem, then you have to talk about what is the input that will be put on the Turing machine input tape.

And that could only be bits and in terms of those number of bits, what is the time complexity, right. So that time complexity then is defined and space is also defined. So we want an algorithm that has time complexity polynomial in the size of the circuit. Like so size actually is now bit size. Is this clear? That is the problem definition, algorithmic problem. And since it is an algorithmic problem, it will not make sense to use constants like $\sqrt{2}$ or e or π.

So we will just use constants. Either they are elements from a finite field or they are elements from the integer ring, right. So we can just use the simplest possible constant. In fact, since you have addition multiplication gates, you can start with just 0, 1, -1 and then using this you can build everything else, all the other constants. So you can also assume constants to be just have three types. So we will focus on the case of R.

The ring will be a field for us. So the field will either be the field of rationals or the finite fields. Now there are innumerable other fields which we are ignoring but it will not matter I mean algorithmic problem will always reduce to one of these two types, okay. So this is really enough to solve. So what is the representation size for constants in s q.

Yeah, so any element in a finite field you can write in just log of the field size, many bits. And the representation for rationals or integers will of course be in bits and it can be arbitrarily big. So how big it is that will be counted in the input. No that is fine, but what the size of a will be the value of a log that many bits. So it can be arbitrary. And yeah, so what is the status of this problem? How do you solve it practically?

So that we have seen in the early assignments, right. This Schwartz-Zippel lemma. So Schwartz-Zippel and there are other authors. So this lemma I mean the lemma you saw in the assignment will tell you that PIT is in CoRP which is definitely in BPP but one-sided error, okay. So it is in randomized polynomial time with one-sided error. The error is on which side?

When the circuit is 0 obviously you will say that circuit is 0 but when the circuit is nonzero then you may be fooled, right. It is on nonzero side. So it is in CoRP. **"Professor - student conversation starts"** Like that there is a problem kind of captures the BPP. Two-sided. BPP is two-sided. Yeah but PIT is one-sided and one-sided problem kind of. Yes that is happening because of PRGs. **"Professor - student conversation ends".**

Yeah. So I think in that lemma about zeros of a polynomial you might not have seen all the cases. So we will just go through the cases without going into the details of each. So you are given a circuit $C(\bar{x})$ of size s. **"Professor - student conversation starts"** What is the status of the problem? Is it contain the rings and not fields? Was not this also part of that question? Okay, it was the domain, but what if it is not a domain. Yeah, so all these things you are supposed to do in that. **"Professor - student conversation ends".**

No, did it not ask for when R is not a field. What happens? Right. Yeah, there is a lot to say. Let us not say it now. Because you have to talk about the structure of the ring and all. Yeah, when you look at the whether it contains a field and then how far is it from the field. So depending on that you can simulate Schwartz-Zippel. Even there the harder I mean, we can pinpoint what are the harder rings.

It is not that for non-domains, the problem is always hard. It is only hard when the non-domain is very far away from a field. Basically we have to, the thing is that the algorithm is different for the two fields two families of fields. For finite field, it is a bit different than when the field is the field of rationals. It has a slight difference, so which is why I am doing this proof.

Because there are some technicalities involved. I mean, idea of course is just that you pick a random point and evaluate your circuit C. But then there are some technical issues in proving, getting the probabilistic showing that this a valid probabilistic algorithm. Yeah F_q may be a small field that is one problem and on the side of rationals the problem will be that at a random point when you evaluate the circuit, the number may become far too big than you can store.

So for example, if you have the degree of the circuit is 2^s, then x^{2^s} if you substitute $x = 2$ there, it will immediately output 2^{2^s}, which you will never be able to store anywhere. So yeah all kinds of technical issues are there. We will see how that is done. So because of circuit having size s so we are in this case of F_q.

You know that the degree of C is bounded by, certainly s^s, yeah it cannot exceed that. In every level you are multiplying s things. So degree cannot go beyond s^s. But this is an exponential degree. So we actually need to apply that lemma of Schwartz-Zippel. We will need that size of the field is greater than s^s, slightly more than s^s. But then that may not be true, the field may be just having two elements.

What do you do then? Yeah, then you go to a field extension. So what will be the degree of the field extension? It will be the log of the size. So that will be $poly(s)$. For this, if needed go to a field extension, degree will not be needed more than s log s or something, $\tilde{O}(s)$. It is not the degree of the field extension is not too large. So do you know how to go to a field extension of a finite field in randomized polynomial time?

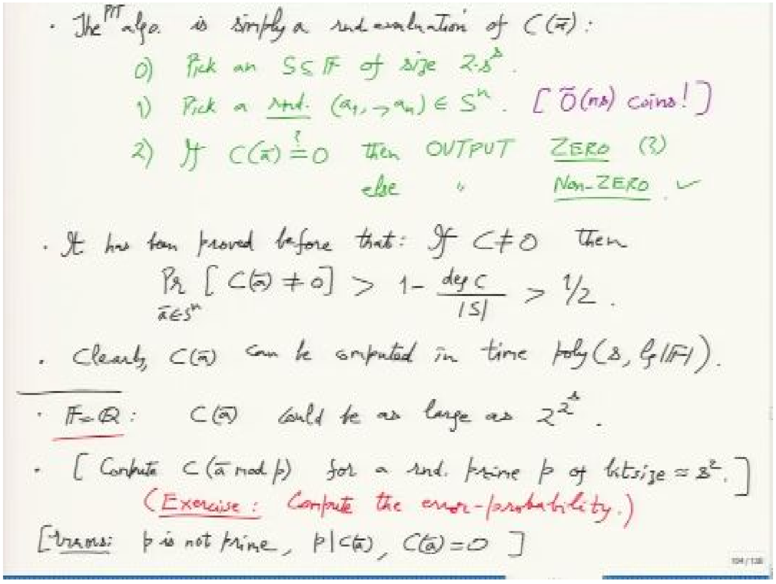
No, I am saying how do you find the field extension of required size? You basically guess or randomly pick a polynomial of degree this much $\tilde{O}(s)$. And if you are lucky it will be irreducible. And that defines our field extension. Well, you have to show that, you have to show why it is picking a random polynomial good enough.

It is possible to do like that, but I will not go into those details. So let me just leave this as an exercise. Finding a field extension. Well so the simple thing is randomized polynomial time, but there is also a deterministic way to do it. So there is a paper by Adleman Lenstra that does it in deterministic poly-time.

So also there is a Adleman Lenstra paper that can be invoked directly from the 80s. This needs a Gaussian sum and exponential sums. So we do not do that in the course, but it can be done. So this actually, if going to a big enough field extension is not a problem at all. It can be done in deterministic polynomial time. Okay, this is the bottom line to remember. More than that you do as an exercise.

So we then assume, basically without loss of generality that our field is quite big. We are so the circuit that we are given over a finite field, that field is exponentially big in terms of the size. And then things are easy.

(Refer Slide Time: 34:39)



- The PIT algo. is simply a rnd evaluation of $C(\overline{x})$:
  0) Pick an $S \subseteq \mathbb{F}$ of size $2 \cdot s^s$.
  1) Pick a rnd. $(a_1, \ldots, a_n) \in S^n$. [$\tilde{O}(ns)$ coins!]
  2) If $C(\overline{a}) \overset{?}{=} 0$ then OUTPUT ZERO (?)
     else " Non-ZERO. ✓
- It has been proved before that: If $C \neq 0$ then
  $$\Pr_{\overline{a} \in S^n} [C(\overline{a}) \neq 0] > 1 - \frac{\deg C}{|S|} > 1/2 .$$
- Clearly, $C(\overline{a})$ can be computed in time $\mathrm{poly}(s, \lg |\mathbb{F}|)$.
- $\mathbb{F} = \mathbb{Q}$: $C(\overline{a})$ could be as large as 2^{2^s}.
- [Compute $C(\overline{a} \bmod p)$ for a rnd. prime p of bitsize $\approx s^2$.]
  (Exercise: Compute the error-probability.)

[Errors: p is not prime, $p \mid C(\overline{a})$, $C(\overline{a}) = 0$]

104/138

So now for this case the algorithm is simply the PIT algorithm. It is simply an evaluation, it is a random evaluation of the circuit. Pick a subset of the field of size two times s^s. Since it is a finite field, you can pick the whole field as is, just take s^s in the extension and pick a random point a $(a_1, \cdots, a_n) \in S^n$. This random process will not be very expensive because how many random bits would you need?

This is $n \cdot \log s$. And the size of big S is small s^s. So it is just $\tilde{O}(ns)$. This just requires $\tilde{O}(ns)$ coins. So you just need to flip these many coins and unbiased coins and it is fine. You have now this point $\overline{a}$. So evaluate C at $\overline{a}$, check whether it is 0, the value is 0. If it is 0, then you will output that C was 0. That is the output.

If C is not 0, well then you have a certificate of the nonzeroness of C. So this is correct and this may or may not be correct, because you answered too soon. I mean this $\overline{a}$ may be a root of C, it may be a 0 of C. So that is what we have shown. We have done an analysis in our previous assignment. So it has been proved before that if C is not identically 0, then the probability that C at a random $\overline{a}$ from this big enough S^n, is quite high.

Okay, so this is $1 = \deg C / |S|$. $\deg C / |S|$ is around half, so this is greater than half. So this output 0 you will be correct with probability 51% for example. And then you can repeat this and you can boost it to arbitrarily close to 1. You can boost it simply because this is one-sided error. So it will be easy for you to see that just repeating this and taking a majority vote brings the probability success probability close to 1.

That makes it a valid probabilistic algorithm and time complexity is clearly poly-time. So clearly $C(\overline{a})$ can be computed in time $poly(s, \log|F|)$. So the input was the circuit of size s and field F, which is a finite field. So in the input size this is a polynomial time algorithm and probability is bounded. So it is probabilistic or it is a randomized poly-time algorithm.

The other case is now F equal to Q. So in this case, $C(\overline{a})$ could be as large as 2^{2^s} in value. Because just look at x^{2^s} and plug in $x = 2$. So this is now beyond our storage capacity. So any ideas? How will you modify this green algorithm. So you reduce modulo primes. For that you should be willing to believe that there are enough primes around.

Once you believe that, then if you need, let us say s^2 bit primes, we just randomly pick on a number of s^2 bits and then do a primality test. Then it is a prime with high probability and you go mod that prime. So this number, for example, $2^{2^s} mod\ p$, where p is s^2, this computation you can do in the circuit.

When you are multiplying and adding you can do that arithmetic mod p at every point. So the size will never, I mean the bit size will never blow up. Because every time you are dividing and taking the remainder, so remainder mod p will only be p - 1 at most. So your answer will be s^2 bits instead of 2^s bits. You have to be willing to believe that, that there are enough primes.

He is not asking about primality test. So it is for s^2 bits, it is $1/s^2$. You just repeat, so your success probability is $1/s^2$. But then you can boost it. You can boost the success probability. The whole day you keep picking primes. And then the probability will be higher that you hit a prime in some hour.

And when you hit then the primality test will also tell you with confidence, very close to 1 that it is a prime. Or you can use AKS which will tell it with 100 percent. So either way, okay. That is the idea. You reduce $C(\overline{a})$. Well do not reduce in the end because then it will be too late. You have to compute modulo p. So compute $C(\overline{a}\, mod\, p)$.

So this computation mod p in mod p arithmetic. For a random prime p of bit size around s^2 let us say. And this you can randomly pick. The probability analysis we will not do. This is again an assignment. You have to use the prime number theorem, but prove it. So that I am skipping. Abhibhav at least has seen in a class. So you are claiming that they can do this exercise.

"Professor - student conversation starts" No what is the probability that it is a prime. That you have to prove. Because the range is exponential probability is $1/poly(s)$. Why is that? That you have to prove. Because you are picking prime in range n_1 and n_2, both are exponential in s. So prime is exponential bit size is poly. What if in that range there is no prime, zero prime.

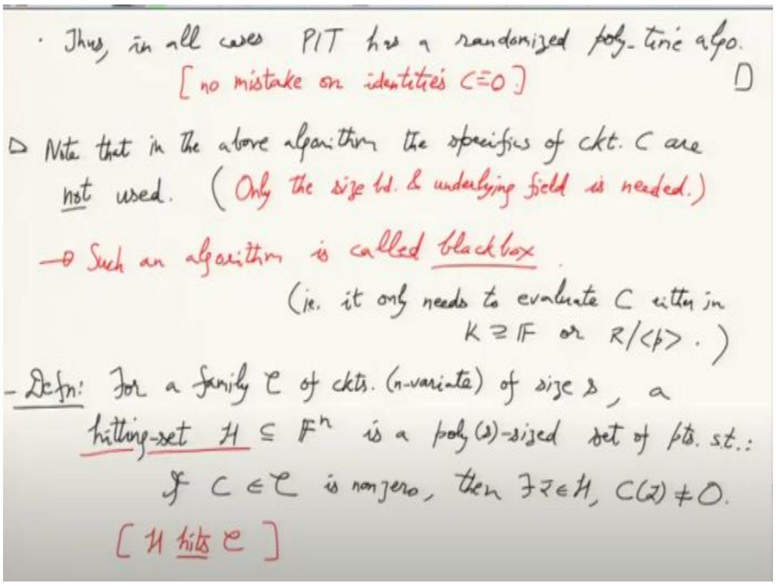
Everything is composite. But what is the proof? There are proofs. Yeah. So those things I am leaving as an exercise. **"Professor - student conversation ends".** So compute the error. So compute the error probability of this process which will give you a prime, but then one minor complication is that this prime may actually divide $C(\overline{a})$. $C(\overline{a})$ is a large number but the random prime that you picked after such hard work of one day turns out to be a factor of $C(\overline{a})$, and so $C(\overline{a}) = 0$ mod p.

So that error probability also you have to estimate. You have to estimate that, the errors can be, p is not prime or p is a prime but then p divides $C(\overline{a})$ and $\overline{a}$ itself may be a root. So there are three types of errors and you have to estimate each of them. $C(\overline{a})$ that we have estimated before.

Prime p dividing $C(\overline{a})$, this you will rule out or you will estimate the probability by saying that not too many primes can divide $C(\overline{a})$ because it can be as big as something like s^{s^s}. And how many prime factors will it have, at most s^s. So these s^s are the bad primes. You remove them. Still s^2 bit primes are way too many. So this is a very miniscule set of bad prime.

So you will not pick them with extremely high probability they will be missed. And p is not prime or p being a prime, for that you need the prime number theorem. These things I leave as an exercise. So if none of these errors happen, then you have a certificate of C being nonzero, with high probability.

(Refer Slide Time: 48:14)



- Thus, in all cases PIT has a randomized poly-time algo.
  [no mistake on identities $C\equiv 0$] □

▷ Note that in the above algorithm the specifics of ckt. C are not used. (Only the size bd. & underlying field is needed.)

→ Such an algorithm is called blackbox.
(i.e. it only needs to evaluate C either in $K \supseteq \mathbb{F}$ or $R/\langle p\rangle$.)

- Defn: For a family $\mathcal{C}$ of ckts. (n-variate) of size s, a hitting-set $\mathcal{H} \subseteq \mathbb{F}^n$ is a poly(s)-sized set of pts. st.:
If $C \in \mathcal{C}$ is nonzero, then $\exists \bar{\alpha} \in \mathcal{H}$, $C(\bar{\alpha}) \neq 0$.
[$\mathcal{H}$ hits $\mathcal{C}$]

So thus in all cases PIT has a randomized poly-time algorithm. And there is no mistake on identities. The algorithm by nature will never make a mistake if C was 0 because it is simply evaluating. In some rings it is evaluating. So if it was 0 it will remain 0 but when it is nonzero then it makes mistakes. This is a practical algorithm, in practice it will run extremely fast.

And the other nice property it has is that it does not care about the circuit. So it takes the circuit as a black box. Just ask the black box to do arithmetic in this ring. You have to request the black box. But then black box will not tell you how it is implementing that. It is addition multiplication, there is no arbitrary process. It is not a neural network. No we are not looking, we are requesting the black box to do arithmetic mod p.

So suppose you are the black box. So you will not show me the wires and the edges and the nodes. You will do the computation and you will just tell me the answer. Because every step in the black box is defined mod p. So you can make this request, it is a valid request.

"Professor - student conversation starts" The black box circuit that comes over p right but it also takes into the prime as well where it allows you to increase the prime. So that will be a formal way to define the black box. Yes. **"Professor - student conversation ends".** The black box will not only compute a polynomial with rational coefficients, but also modulo primes.

So you give the prime. Nowhere in the analysis did we use anything about the circuit looking in a particular way. That is what black box means. **"Professor - student conversation starts"** Is there any difference in like any setting where it matters whether you consider the constants as part of your sets or not. No, o it will only matter when the coefficients are integral because they can be arbitrarily huge. **"Professor - student conversation ends".**

When you are leaving it to the black box so how will the black box compute mod p a number that is triply exponentially large. That but so if that is being done by the black box then you. **"Professor - student conversation starts"** Instead of requesting the black box it will come with mod p. If we just ask the black box to tell you whether it is 0 or not and then the bit size does not matter.

But when you are talking about circuits being given to you then you have to talk about input size. **"Professor - student conversation ends".** Then you know that the coefficients are only exponentially large. They cannot be triply exponentially large. I am not saying they will not work but there are algorithms and it is also conceivable that you can use the constants and the wiring.

In black box model all you know is the circuit size. That is a property of the box. I mean, we usually think of the number of variables also in the size. So s is the size bound and there are s variables and there will be a single output. No, in this the theorem was not talking about black box at all. Theorem was just saying that the problem, algorithmic problem of PIT is in CoRP that is all.

There is no black box here. So that is follow-up observation that in the above algorithm the specifics of the circuit C are not used. It only requires a size bound. Only the size bound and field, underlying field. Such an algorithm is called a black box identity test or black box PIT. Okay, so it was presented as a white box algorithm but actually it can also be used as a black box algorithm because it never cared about the circuit anyways.

It is just evaluation based. It only needs to evaluate C. Now this is where the older question which you were asking applies. So evaluate C where? Black box should allow you to evaluate either in an extension of the ring or in a reduction of the ring. So evaluate C either in extension K of your field or $R/<p>$. Evaluation should be allowed in either extension or reduction.

So yeah it is true that when we talk about black box these are the assumptions. That the black box will allow us to go either to a bigger field or reduce mod p when possible. It may also be another, you can also say reduce modulo. So you substitute for example, $x_1 = t$ and say that I want the answer mod t^2. Those other ideas can also be used but usually they will all reduce to these two cases.

Nothing stronger happens by I mean except this case. So yeah, so if this is confusing, if this model is confusing, there is a much better way to talk about black box PIT which will be by removing the circuit C from the picture. Now we will not even look at C, we will not look at the black box. Our question will just be a design question designing a set of points okay without looking at C at all.

You can think of that as a non-adaptive algorithm. You are evaluating the black box at different points, but your next result does not depend on the previous result or next evaluation point does not depend on the previous results. So we call that set hitting set. For a family C of circuits, let us say n-variate of size s a hitting set H is a subset of F^n.

It is a poly s sized set of points such that if a circuit C is given in the family C which is nonzero then, then what? What should the hitting set do? It should have a non-root of C. So then there exists an $\overline{\alpha}$ in the hitting set such that $C(\overline{\alpha}) \neq 0$. We say that the set H hits the circuit family C.

Because whenever you pick a nonzero circuit in the family there is a non-root in H, and you should notice that the circuit family has, even when you fix the size s and n usually there will be infinitely many circuits simply because the field is infinite. So for these infinitely many circuits, there is a small hitting set. At this point, you should wonder whether these powerful things even exist.

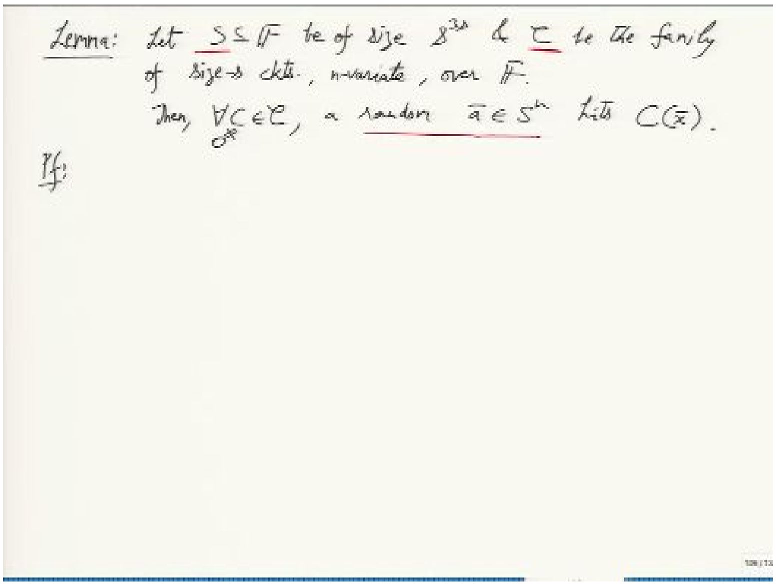
Because infinitely many polynomials will have infinitely many roots. And why are they away from H. H is a small set. So there is really no reason at this point. So colloquially we will say that H hit C. Now it is a design question without even looking at our circuit. The question is just, you will be given let us say this number is in unary. On the Turing machine on the tape, s will be written in unary.

So 1^s is input and you have to output in the output tape of the Turing machine a set of points H. And the question is how fast can you do this. That is contained in C. That is a description of C. But the Turing machine question or the algorithmic question is just given 1^s output a set of points and this algorithm should be as efficient as possible.

And you do not even know whether this H even exists at this point, forget about the efficiency. You can think of C as a kind of VP just any circuit is given to you. Or you can think of it as depth-4 or you can think of it as depth-3 or depth-2. So these are the example settings we are interested in. What about the existence question? Say I remove this condition of $poly(s)$ sized.

Then can you at least show that there is a finite sized H that will hit all these infinitely many circuits. So that is Schwartz-Zippel. So another corollary of Schwartz-Zippel is that there is a finite H. Hitting sets exists, finite hitting sets exist. Finite and inefficient hitting sets exist.

(Refer Slide Time: 1:03:03)



Lemma: Let $S \subseteq \mathbb{F}$ be of size s^{3s} & $\mathcal{C}$ be the family of size-s ckts., n-variate, over $\mathbb{F}$.

Then, $\forall C \in \mathcal{C}$, a random $\bar{a} \in S^n$ hits $C(\bar{x})$.

Pf:

So let us just state that lemma and then we leave. So let S be of size s^{3s} and C be the family of size s circuits n-variate over, I do not need a finite field. I think just over F. So S is a large enough set and C is a family of all circuits size s. Then a random $\overline{a}$ in S^n hits C. This is just a reformulation of Schwartz-Zippel.

I mean this is not weaker. No, but why random, random $\overline{a}$ hitting the probability is hidden here. So this is Schwartz-Zippel. What is the hitting set that this is giving, S^n. So S^n is your hitting set. Because I mean hitting set just needed one element $\overline{a}$ and here almost everything in S^n will hit your circuit. Actually no. I see.

I wanted to say something else actually. Let me change it a bit. let me then for all C in the family a random $\overline{a}$ hits the circuit. No, there is also a formulation with a single point, but let me specify the circuit here. So for any nonzero circuit C in the family, random $\overline{a}$ will be able to hit it. This is exactly Schwartz-Zippel.

If you want the previous result, a single point that hits every circuit in the family, then I will have to talk about finite field here. Because if the field was finite, let us say of size q, then the number of circuits is only q^s. So the circuits are actually few. In Schwartz-Zippel you do a union bound. You will get $q^s \cdot s^s/s^{3s}$. Then actually you can also do that.

But so these are simple implications from Schwartz-Zippel. They still do not tell you whether there is a small hitting set. Actually the one point was fine. The hitting set of size 1 exists. That was okay. But it does not tell you anything about efficiency. How will you produce this hitting set? So let me stop now. So next time we will continue the study of hitting sets.