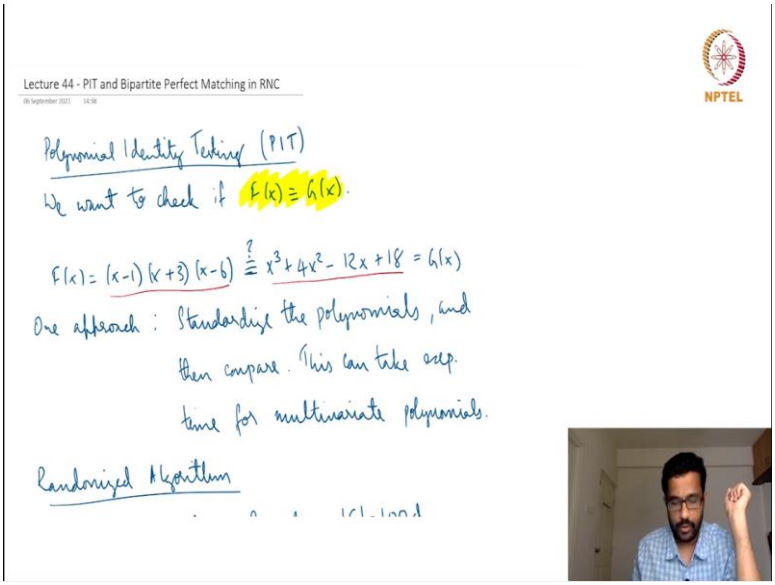
Computational Complexity
Prof. Subrahamanyam Kalyanasundaram
Department of Computer Science and Engineering
Indian Institute of Technology, Hyderabad

Lecture - 44
Polynomial Identity Testing and Bipartite Perfect Matching in RNC

(Refer Slide Time: 00:15)



Hello and welcome to lecture 44 of the course computational complexity. So, today in this lecture we will see a connection between randomized algorithms and circuit complexity. So, we will see this language which we had briefly seen during when we talked about a randomized algorithm called polynomial identity testing and short PIT. So, what is polynomial identity testing? We will see that.

And we then we will see an application to of polynomial identity set testing to a problem of determining whether a given bipartite graph has a perfect matching. And then that would imply that the problem of determining whether a bipartite graph has a perfect matching this problem is in a class called randomized NC. So, polynomial identity testing simply is asking if, given 2 polynomials F and G it simply asking whether F and G are the same polynomial.

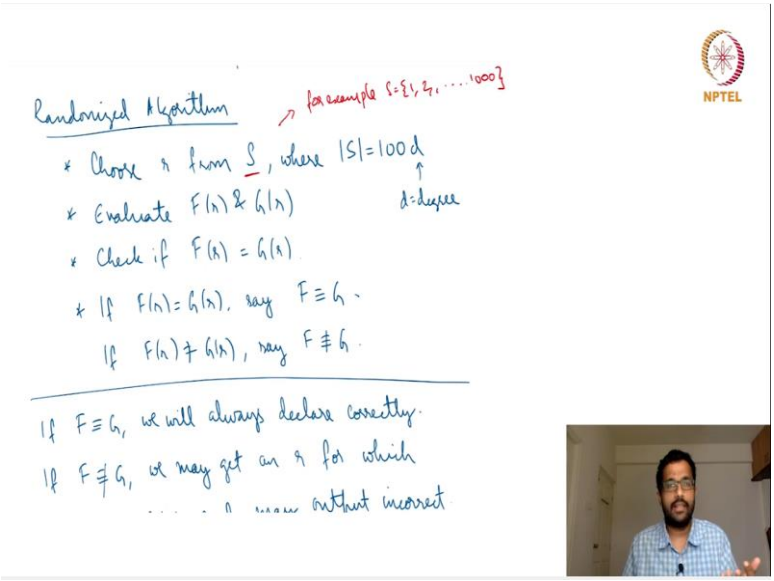
So, one may think what is the big deal to just compare the 2 polynomials. The catch is that these 2 polynomials need not be given in the exact same format. If there is a standardized format like such as the leading term has the highest degree then the second term has a second

largest degree or some standardized format. Then it is just a matter of checking between the 2. But then even the way the polynomials are specified itself could be in different ways.

So, this may require us to do some computation. So, just to give an example what if the polynomials are F and G are given like this x - 1 + multiplied by x + 3 multiplied by x - 6 and like this x cube + 4x squared - 12x + 18 this is G. So, now how would you do this? So, you can try to reduce the polynomial to the standard form. But on the first look both of these the leading terms look like x cube so, that seems to be same.

The constant term here is +18 in the left hand side and the right hand side is +18 left hand side it is - 6 times + 3 times - 1 which is also + 18. But then when you look at the coefficient of x squared it will be just the sum of this - 6 + 3 - 1 so it will be - 4 whereas in the for G it is + 4 so you see that these are not the same.

(Refer Slide Time: 02:58)



So, one approach is to standardize the polynomial and compare and this can be done for univariate polynomials. So, univariate means polynomials of a single variable such as this here the only variable in F and G above is x. But if you have multiple variables let us, say the number of variables is n then potentially you could have you could have many such factors like x 1 + x 2 multiplied by x 3 + x 4 multiplied by x 1 + x 6 multiplied by x 1 + x 7 some lot of terms.

And when you try to multiply them all it could result in an exponentially large polynomial. So, the given polynomial may be in a very crisp short format but when you try to expand

them it could result in an exponentially long polynomial. And this will make our algorithm non-polynomial time. So, this will lead to a non-not inefficient algorithm. So, this is not something we want to do.

So, this problem for multivariable polynomials in particular can be hard. And in fact, as of now as of today, we do not know an efficient algorithm for check for doing for this problem polynomial identity testing then the input polynomials are multivariate. So, when the input polynomials are single variable then it is easy you could just x 1 x - 1 into x + 3 you could just multiply them out and this can be done efficiently but not so in the case of multi multivariate polynomials.

So, let us try a very simple randomized algorithm. I think I touched upon this while discussing randomized algorithms myself but still I will just quickly go through it. So, choose a random number in place of x. So, let us say we choose r from and instead of saying choose a random number without any without specifying from there let us say we choose a random number from a fixed set S.

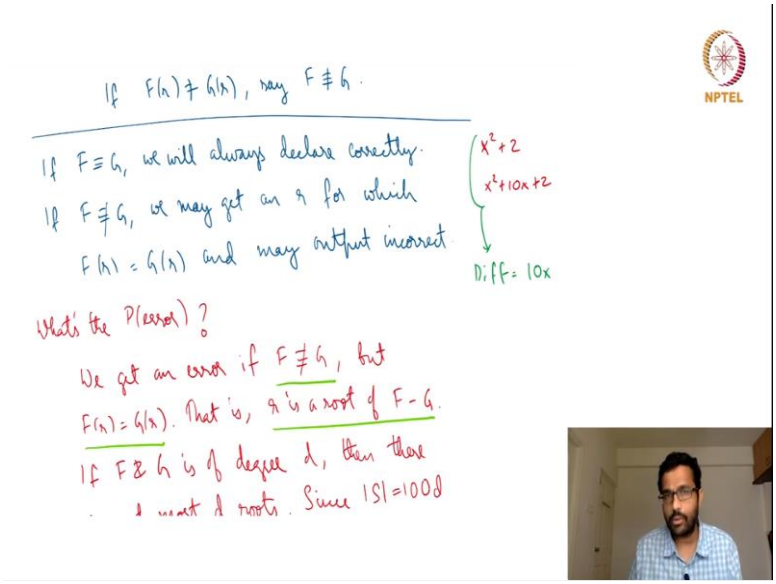
So, this could be some so say let us say it is degree 10 polynomial so S could be some set of size thousand. So, you could say S for example x s, s could be this is the first thousand natural numbers. So, S could be anything now you and from that S you randomly pick a number r. Now you evaluate the polynomial F so it is one thing to simplify the polynomial it is another thing to evaluate it.

So, now suppose r is let us say 10 right so that for G here you have 10 cubed + 4 into 10 square - 12 into 10 + 18 so 1000 + 400 - 120 + 18 you can compute that. For F it is just 10 - 1 9 10 + 3 13 10 - 6 4 so 9 times 13 times 4 it is a product. So, it is one thing to evaluate and one thing to expand without evaluating. So, just evaluate the polynomials at r both the polynomials and then check whether they are the same.

And if the evaluations happen to be the same you say the polynomials are equal. And if the polynomial are identical if the evaluations are different then you say the polynomials are not identical. So, this is the product this is a very simple randomized algorithm in fact there is some one thing that you will keep seeing whenever you encounter randomized algorithms. The algorithm that you see usually will be extremely simple to describe and program.

But the proofs may be long the proof of correctness or proof of running time this could be long but the problem the algorithm itself is usually very simple so how simple how simpler could it have been. You have part two polynomials that you want to determine whether they are identical you just substitute random values and check whether they are the same they evaluate to the same value.

(Refer Slide Time: 07:03)



So, now the correctness let us see the correctness. Suppose let us say you when you evaluate the polynomial at 10 let us say now F of 10 is different from G of 10. This necessarily means that the polynomials are different because if the polynomial were the same then you evaluated any x, they should give the same evaluation. So, if F and G are this are the same then at any point r, they will always give the correct same evaluation.

And therefore, if F and G are the same you will always output the correct answer, we will always say it is identical. However, if F and G are not the same, in that case there is a small possibility that you may end up getting an r for which the polynomials could be different but the r led you to saying that the r force you to say that F and G are the same even though they are different. So, just as simple example maybe let us say if you have x squared.

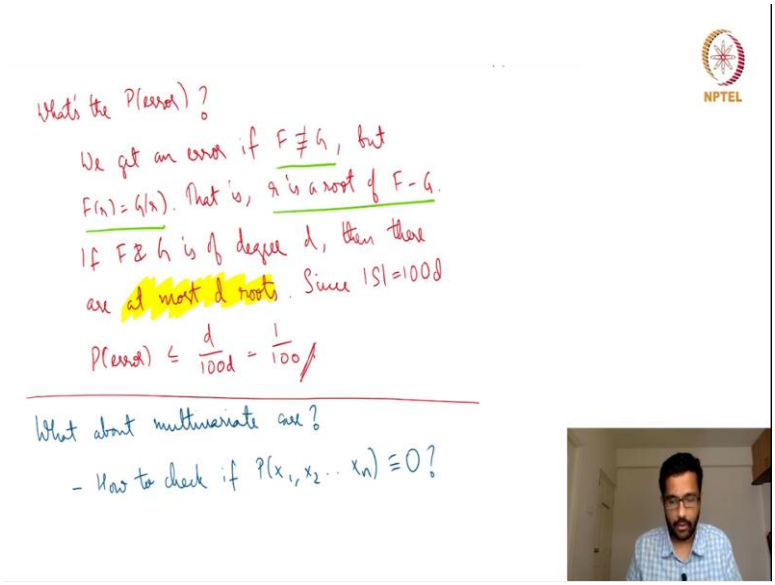
And let us say x square + n x that is x squared + 2 n x squared + 10 x + 2 at x = 0 this these 2 these 2 polynomials are different x + x square + 2 n x square + n x + 2 but at x 6 = 0 you see that they evaluate to both evaluated 2. So, this is an example of a case so which are the r for

which fn. So, the point is that there may be r for which F and G get evaluated to the same value and this is the only possibility of error.

So, even though the polynomials are the same it could so happen that you pick an r for which they evaluate to the same value. So, what is the probability of error the probability of error happens when only when F is not equal to G or F is F and G are different polynomials. When F and G are the same polynomial there is no error. But the error happens when F and G are not equal but Fr and G r are the same.

And this happens when r is a root of the difference polynomial the difference is F – G. So, here what happened here x squared + 2 n x square + 10 x + 2 the difference is so the difference is simply x squared cancels two cancels it is just simply the polynomial 10 x. So, here the choice x = 0 happens to be a root of ten x that is what happened here.

(Refer Slide Time: 10:00)



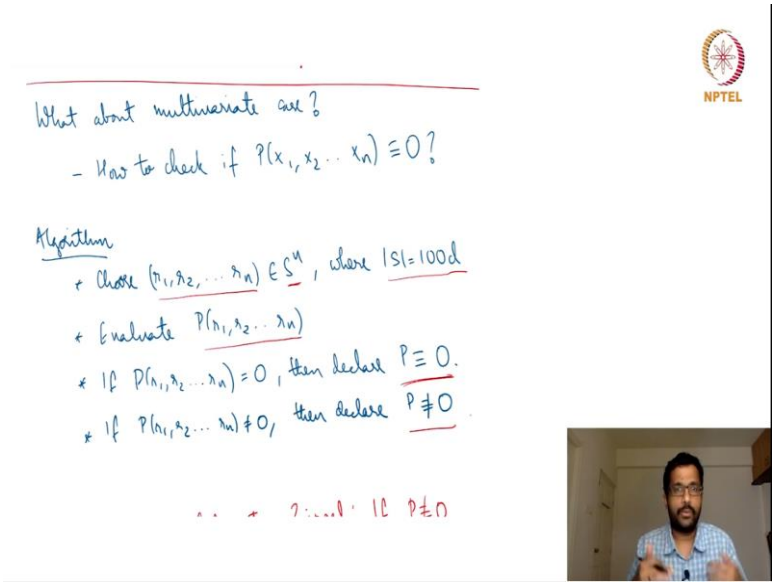
And it could happen in different ways so r is a root of F - G but we have what is called the fundamental theorem of algebra which says that if a polynomial is of a certain degree, then it lets a degree d then it has at most d roots. So, if F and G are of degree d then the difference is also degree d at mostly maybe even smaller and in that case the difference has at most d roots and you and you picked r from a big enough set.

So, if you recall we picked r from a set of size hundred d. So, there were 100 d values and out of which you chose one value and there are at most d roots. So, not all of these d roots need not all of these d roots may be present in the set S but the maximum number of roots that can

be present is all of them. So, and that is d of them so and we give the wrong output you report the wrong answer if r happens to be one of these d roots or one of these roots.

So, the probability of error is picking one of these d roots from the set of size 100 d so which is d divided by 100 d which is at most 1 over 100 of course you could boost this probability of error. So, this is a very simple randomized algorithm that for the polynomial identity testing. But still what I have described is still single variable case and I already told you for single variable you could simplify the polynomial. And then get the answer directly instead of all this substituting or evaluating at a certain point.

(Refer Slide Time: 11:58)



For the multivariate case so let me just say so instead of using F and G let me just use one polynomial P because if F is the same as G, then the same as saying F - G = 0. So, in in instead of that we will just check whether the polynomial a polynomial is identically equal to 0, identically equal to 0 means the polynomial should be the 0 polynomial each monomial should have coefficient 0.

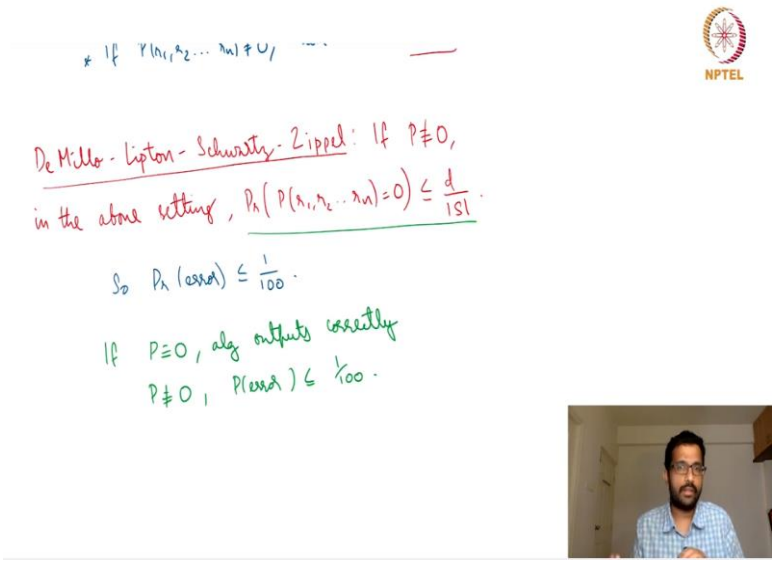
So, it is just a 0 the polynomial is itself 0 not that it gets evaluated to 0 at some point. It is 0, it is everything is 0. Now again this may seem straightforward but then we may have some strange ways of accessing or the polynomial may not be given to us in a standard format. It may be given to us an addition multiplication difference everything or it may it is also possible it is given to us in some query format.

So, again we could do the same kind of idea right so now we the only difference is that we have n variables x 1 x 2 up to x n instead of 1 variable x, so similarly now we pick not just 1 r but r 1 r 2 for each of these x 1 x 2 etcetera. So, we pick r 1 r 2 etcetera from S power n where S is of size 100 d the same set as a same or similar sized set. So, that is r1 is chosen from S r 2 is chosen from S etcetera at random.

And then you evaluate the polynomial at these points r 1 r 2 up to r n. If it outputs if the polynomial at r 1, r 2 up to r n evaluates to 0 then you declare that the polynomial is the identically the 0 polynomial. If it evaluates to a non-zero value then you declare polynomial is not identically 0. So, the correctness is I hope we are fairly clear or at least to the problem with the exception of the error.

If the polynomial is the 0 polynomial, then whatever r 1, r 2 you substitute will give you 0. So, in that case you will always output 0 the polynomial is 0. But even when the when the polynomial is not the 0 polynomial there is a possibility of r 1, r 2 etcetera being such or being chosen in such a way that or again the charges are all random but then you could happen to choose the r 1, r 2 etcetera in such a way that the evaluation is equal to 0.

(Refer Slide Time: 15:04)



So, there is a probability of choosing or of unluckily choosing the bad r 1, r 2 etcetera which evaluates 0 even though the polynomial itself was not the 0 polynomial. So, you may incorrectly output that the polynomial is 0 when the polynomial is not 0 that is the only possibility of error. In the case of single variable, we were able to say something like this we were able to say; these if F and G are of degree d then there are at most d roots.

In the case of multiple variables can we say something like that. In the case of multiple variables can we say it has at most d roots or something like that. That is the question if you have if you can say that then we are done. So, we cannot exactly say that but we can say something very similar and that happens to be enough. So, these is the short zipper lemma or also sometimes are also called DeMello Lipton Schwartz Zippel lemma.

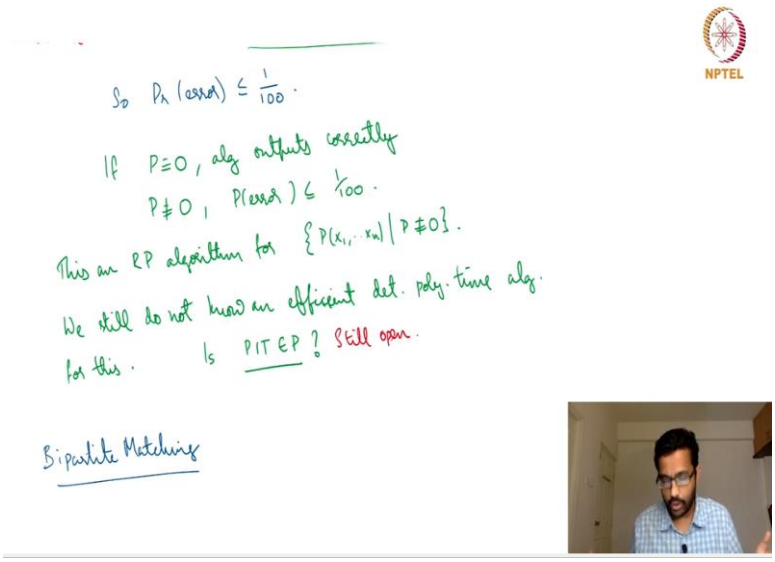
Which says that if the polynomial is not identically 0 which means it is not the 0 polynomial and in this above setting where we choose S from a set of 100 d where d is the total degree of the polynomial. Then the probability that you evaluate to 0 you pick r 1, r 2 and these r 1, r 2 evaluates the polynomial to 0 this is at most d divided by the size of S. So, this is exactly like what we wanted about.

So, here we said the probability of error was d divided by the size of S which was d divided by 100d. And again, the same thing here the probability of error is d divided by the size of S. So, the probability of error in this setting is 1 divided by 100. So, if P is identically = 0 algorithm outputs correctly and if P is not identically = 0 polynomial probability of error is at most 1 divided by 100. So, there is some probability of error and this is at most 1 divided by 100.

So, this is the so strictly speaking if when you are looking for identity testing meaning whether this is 0 or whether the polynomial is 0 or not. That is why it is called identity testing meaning identity is means 0 is the identity polynomial meaning it is additive identity. This is a co RP algorithm when you look at it in that perspective. But if you want to change the language around if you are checking whether the polynomial is not identity then it is an RP algorithm.

So, again RP co-RP or complement if l is in RP, then l complement is in co-RP. So, this is a RP algorithm.

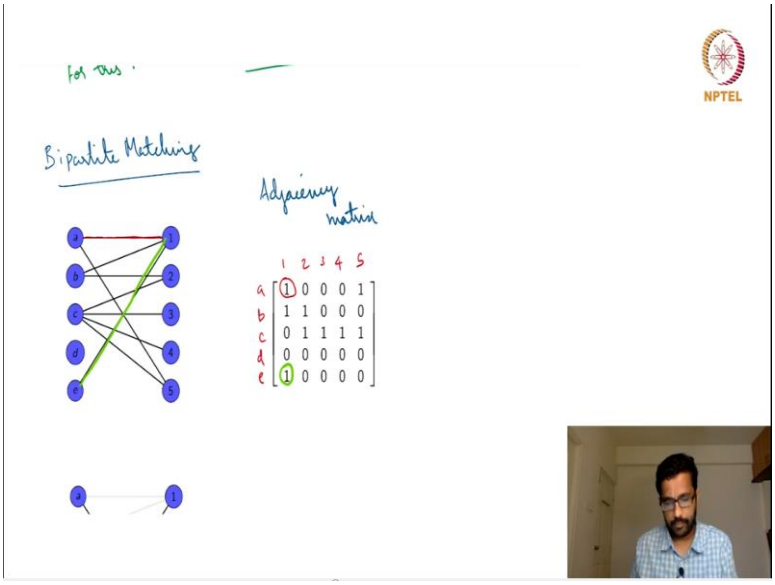
(Refer Slide Time: 18:29)



If you are looking at the question of so this is an RP algorithm for the language not for identity testing for, I will just write down the language set of all P let us, say x 1 up to x n P is not identically 0. So, there is a small error in the case of it not being 0 and when in the case of its 0 it is outputting correctly. And this this problem which is this is kind of interesting and the reason I am say saying this is because so we saw a simple RP or co-RP randomized algorithm.

We still do not know an efficient deterministic polynomial time algorithm for this. In other words, is PIT in P this is still open. This is one question where randomized algorithm seems to help. We know of a randomized efficient randomized algorithm but do not know still whether there is a deterministic polynomial time algorithm. So, this is one of those problems where this this is the case.

(Refer Slide Time: 20:25)

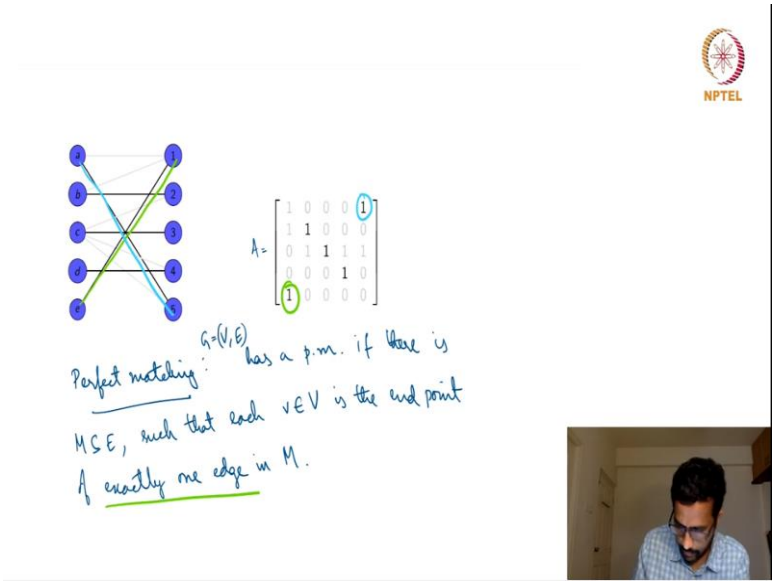


Now I hope this this explanation is clear. Now I will move to an application the application of bipartite matching and we will that will lead us to seeing that the problem of bipartite perfect matching is in the class r and c. So, a bipartite graph is a graph where all the vertices can be classified into 2 groups partitioned into 2 groups and all the edges are cross edges going across. So, like in the picture over here a b c d e are on the left side 1 2 3 4 5 are on the right side.

And all the edges are going across and you can write a adjacency matrix. Where so the usual adjacency matrix writes all the vertices on one side and it is the number of rows and columns is equal to the number of vertices. But for bipartite graphs for undirected bipartite graphs even for undirected bipartite graphs you could have a somewhat more you could compress the adjacency matrix into the following form because there are no cross edges.

So, what I mean is you could the rows are indexed by a b c d e and the columns are in index by 1 2 3 4 5 and from a to 1 there is an edge. So, for a to 1 there is this edge and that is this one that I have circled right and from e to 1 there is another edge. And that is the edge I have circled here. So, like that you could you could see how this relates to how this adjacency matrix relates to the graph given, so this is the adjacency matrix of the bipartite graph.

(Refer Slide Time: 22:43)

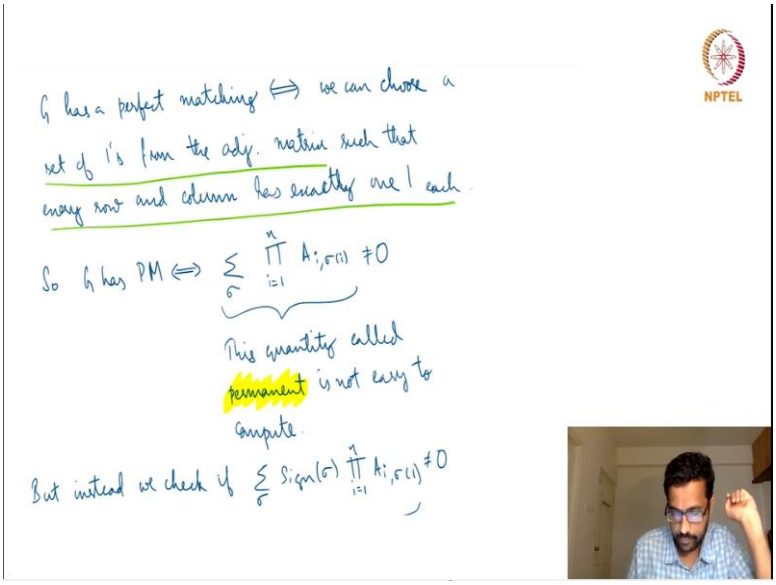


So, the question is there a perfect matching. What is a perfect matching? A perfect matching is a subset of the edges such that each vertex appears exactly once in the; amongst the edges. So, maybe I will just write that down also, any graph has a perfect matching so I just say p m if there is m which is a subset of edges. So, maybe I will just say G = V, E. So, V is a set of vertices and E is a set of edges.

So, if there is M there is a subset of edges such that each v in V is the end point of exactly one edge in M. So, each vertex must appear as the end point of exactly one edge in M. So, this is exactly is important it is not at most it is not at least it is exactly one edge in M. There and for the above graph we have chosen a perfect matching here, so appears one b appears one c appears one d appears one everything and same for 1 2 3 4 5.

And correspondingly I have highlighted the ones in the adjacency matrix that corresponds to these edges so a 5 so the e 1 is this this one and a 5 is this one and the other ones you can see.

(Refer Slide Time: 25:04)



So, this is a perfect matching and this is we are asking whether the graph G has a perfect matching. So, and if there is a perfect matching what I am saying is that now if you look at the ones in the adjacency matrix, I will just erase this over here if you look at the ones you can see that every row has exactly 1 1. Every column has exactly 1 1 so there is no row that has 0 1 or more than 1 1 every row and column has exactly 1 1.

So, G has a perfect matching if and only if there is a we can choose a set of ones from the agency matrix which is a one from the adjacency matrix is that every row and column has exactly 1 1 each. So, now I could write the same thing so this is what we will try to encode so recall the idea is to make sure that perfect matching is in randomized NC. So, we will try to we are trying to move towards that so randomized NC is still a circuit class.

So, one way is to is to look at what is called the permanent? So, permanent basically checks so now you look at it so for row one it is a fifth in the in this matrix. If you look at it the row 1 it is a fifth entry that is the column fifth column for row 2 is the second column row 3 the third column row 4 is the fourth column and for row 5 is the first column. So, each row is matched to something each column is matched to something.

So, and all these entries should be 1. So, there should be a rearrangement of the there should be a remapping of the rows a mapping of the rows to the column such that all the entries of the corresponding things are ones. So, there should be a sigma such that A i, sigma ith entry is 1 for all the i there should be permutation such for all i there is a sigma ith column. So, that the i, sigma ith entries are all ones. So, this is what we want to check.

So, we will add so there is a summation here over all the permutations sigma such that if you take the product of all the entries A i sigma i. So, once the sigma is fixed you are just so sigma could be the identity permutation. So, if you take the identity permutation here it will be 1 1 2 2 3 3 4 4 and 5 5. But that will be 1 times 1 times 1 times 1 times 0. So, the identity permutation will give you the product 0.

If you take some other permutation let us, say 1, 2 it will be this 1 2, 3 3, 4 4, 5 and 5, 1 again it will result in the 0 product because there is only 2 ones. But if you took the permutation that corresponds to the perfect matching you will get one for the summation. So, this perfect matching the existence of the perfect matching corresponds to there being this quantity not being zero.

And this quantity is actually a well-known quantity of a matrix called permanent. So, again this this name is not important right now but this is just a this is what is called the permanent. And we will see in the upcoming lectures or in upcoming weeks the permanent is not something that is easy to compute. In fact, it has been shown that it is even more harder than like it is as hard as like more harder than NP and things like that more harder or at least as hard as NP.

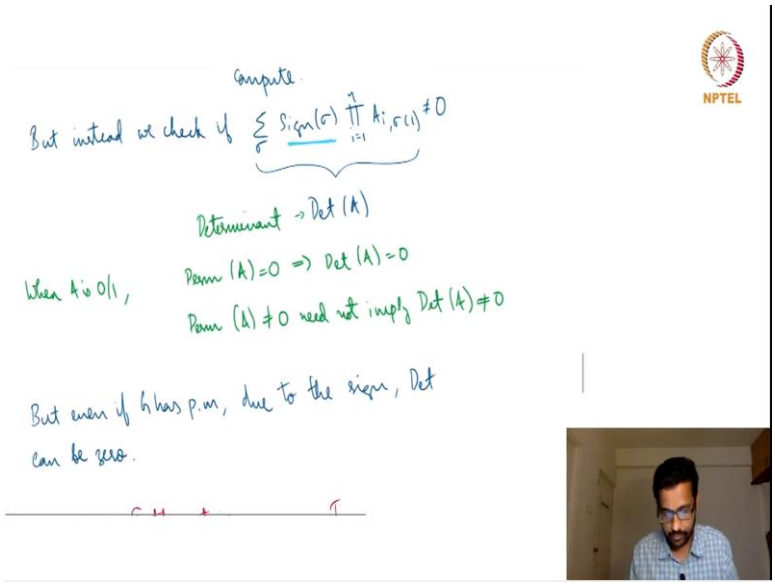
So, this quantity is called permanent and it is not easy to compute so we are taking for all the permutation of for all the rows we are taking the permutations of the columns and checking whether they multiply to 1. So, what is the permanent of this matrix actually the each in fact you can see that you can see this so if you look at the term inside the summation this term is either 1 or 0 for a 0, 1 matrix.

Because it is just a product so it is either 1 or 0. The number of times this term will be non-zero or the number of times this term will be 1 will exactly be the time or the time will be exactly be the times when the permutation corresponds to a perfect matching. So, is the same as asking how many perfect matchings does this graph has and you can check how many this has I think e should map to 1, no matter what and e mapping to 1.

Because that is only edge of e that would mean a cannot get 1 so a has to map to five. So, this graph the top graph is not the same as bottom graph because it d does not have any partner in

the bottom graph. The top graph and once these 2 are fixed a to 5 and b 2 e to 1 then b has to move to 2 and d has to go to 4 so this is only one perfect matching that this graph has.

(Refer Slide Time: 31:04)



And so, for this particular graph this permanent will actually be equal to 1 and in general it will be equal to the number of perfect matchings however I said that this quantity is not easy to compute. So, now what is the way out? In fact, that is a very related quantity to the permanent quantity very similar to permanent that we are very familiar with. So, if in addition to all of this if we included like a sign point a sign here.

Sign meaning if we introduce a sign quantity here then it becomes what we know what we are familiar what is familiar to S as the determinant. So, which is what I have written here so this is exactly the same as permanent but I have just introduced a sign. So, sine is like plus or minus it is just so if the sigma is an odd permutation, it is -1. If sigma is an even permutation is this + 1 so it is just it is just alternating the terms into plus and minus terms.

And then it becomes very familiar to us. So, if you recall the definition of determinant, it is just that 1 into the determinant of this sub matrix. And then minus 0 into the determinant of the corresponding sub matrix plus 0 into this so, there is a definition of the determinant like that. So, this determinant is similar to permanent but it has a signed quantity and but then we know that determinant is easy to compute.

It is kind of counterintuitive because the addition of sine actually we feel makes it plus and minus and makes it harder to compute. Whereas if the top one without the sign it is just all

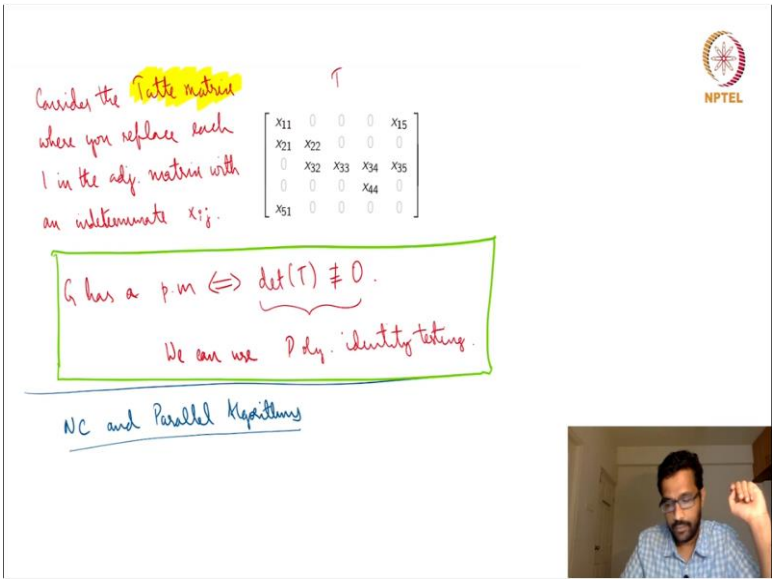
additions, we would think that this is simpler to compute. The b would it would seem that permanent is easier one but interestingly determinant is easier one and permanent is the harder one. So, this is a determinant.

So, the determinant but what we know is that um suppose the permanent is equal to 0 which means all these products should be equal to 0. We are still dealing with a 0, 1 matrix. So, all these terms will be either 1 or 0 so it us just a addition of a bunch of ones so, if permanent is 0 then no matter what you put a sign the determinant will also be 0. However, if permanent is nonzero, permanent of a is 0 implies that determinant of a is equal to 0.

Because all the individual terms must be 0, so in the case of when a is 0, 1 a is 0, 1 matrix. However, permanent of a is not 0 this need not imply determinant of a is not 0. Meaning even if there are some if the permanent is non-zero it could so happen that this because of the sign plus and minus some terms could cancel and determinant could still be 0. So, permanent checking that even though determinant is easier to compute checking the determinant is not the correct thing.

Because there could be matching which means permanent is nonzero but then these matching could cancel with each other and resulting in determinant being 0. But if determinant is non-zero, we know for sure that permanent is non-zero so we know for sure that there is matching. But if the terminal is 0 it could either be because there is no matching or because there are matchings that cancel with each other.

(Refer Slide Time: 35:23)



Because of the sign so, the sign creates this problem. So, what we will do is to we could use polynomial identity testing to check this. So, the problem of cancellation comes when this these are all plus 1 0 and 1. So, what we can do is to take the same matrix and replace the 0, 1 with variables. So, I replace the ones with variables and 0s b 0s. So, this is the matrix 1 0 0 0 1, 1 1 0 0 0. So, now I am replacing each one of them with variables.

So, the first one is x 1 1 the last one is x 1 5 second row is x 2 1 and x 2 2 and so on. All the ones have been replaced with variables. So, they are all different variables they do not have any relation to each other. So, each one is replaced by these zeros remain zeros. Now the claim is that as a polynomial now these do not cancel because each row and each column have different whenever there is one it is a different variable so it cannot get cancelled.

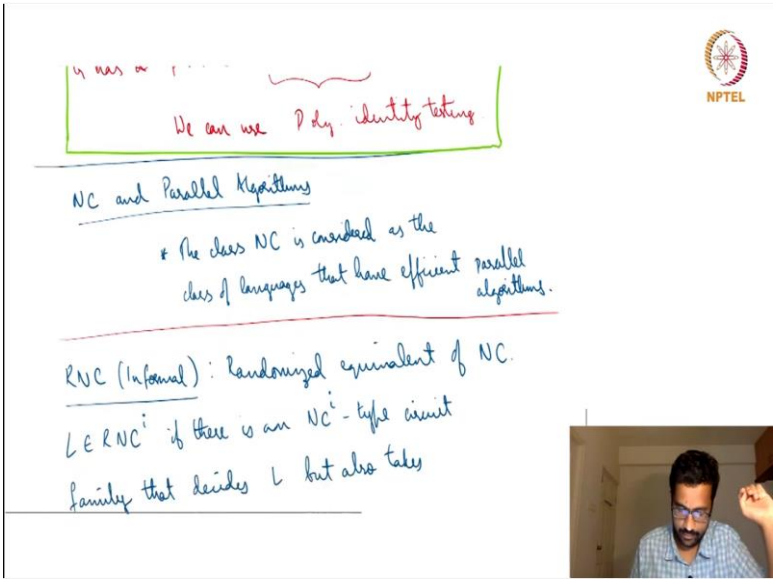
So, even though they are opposing signs you cannot get two terms to cancel with these with each other because the variables involved are different. So, G has a perfect matching right this is the main point if and only if the determinant of this this matrix is non-zero, I said so the determinant is a polynomial it is a polynomial in these variables this this determinant is not identically 0. Meaning it is not the 0 polynomial.

So, just one more point this matrix is called the Tutte matrix t u t t e after the graph theorist and we use the symbol t to denote this. So, we are asking whether this determinant is non-zero is what we are asking here if this determinant is 0 means determinant is not 0 means there is a perfect matching. If the determinant is 0 there is no perfect matching because there is no way that things could cancel determinant 0 means permanent of the adjacency matrix is also 0.

And to test this we could just you say a polynomial identity testing, what we saw earlier. And so that is the algorithm. So, we can use so the graph has a perfect matching if and only if the polynomial is not identity or not identically 0. So, this gives an actually an RP algorithm because if the graph has a perfect matching then the polynomial is not identically 0 and then your algorithm outputs correctly.

So, it is asking this is an RP algorithm for the polynomial for not identically 0 it is an RP algorithm.

(Refer Slide Time: 38:49)



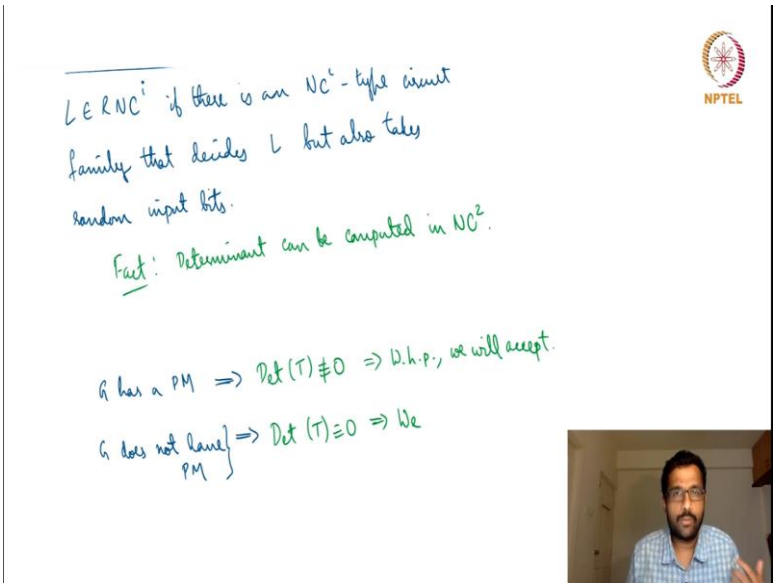
So, now let us see why this implies? So, we are talking about the problem of asking whether the graph has a perfect matching. So, the language is decision version of whether the graph has a perfect matching, it is not asking so we will come to that in a bit. And this implies that this problem is in a class called randomized NC. And why is that? So, before that I will just say one more point.

So, the class NC is usually considered to be class of problems on which we could for which there are efficient parallel algorithms. Because we say NC is like bounded depth so, we could we could consider each gate as a different processor and you have you have bounded depth so, you have n inputs so presumably lot of each input is being processed by a different gate. So, this can be thought of as a parallel process for parallel processing.

So, the class NC is considered as the class of languages that have efficient, I think I mentioned this when we defined or got into circuit complexity at the beginning lecture of circuit complexity efficient parallel algorithms. So, here the point I want to make is that now the question of whether the graph has a perfect matching is reduced into computing the determinant of a certain matrix of the dirt matrix.

And what is the algorithm that we do use to compute the determinant? We will use polynomial identity testing and the algorithm that we described for polynomial identity testing was a randomized algorithm. So, we know we said that our polynomial non-identity testing was a randomized algorithm. So, it is actually a randomized algorithm it is a computation of the determinant but also using randomized randomization.

(Refer Slide Time: 41:30)



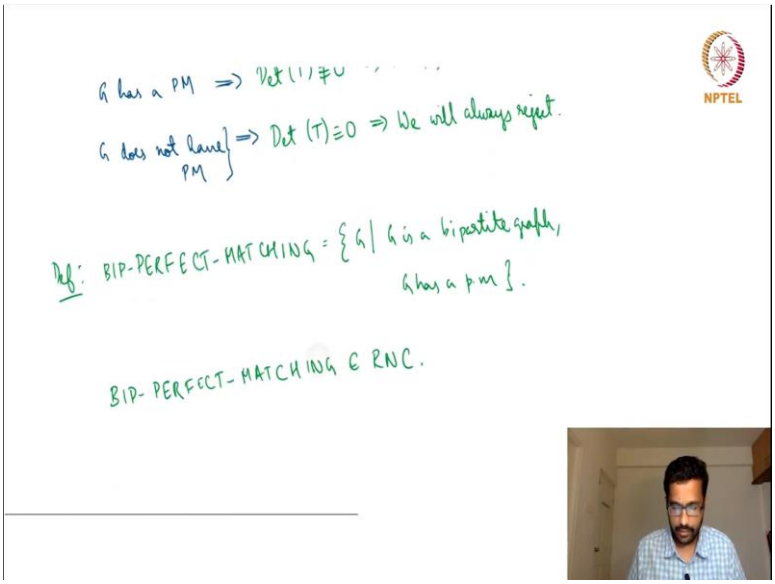
So, the class randomized NC or randomized equivalent of NC sometimes referred to as RNC is the same as NC in in the sense in the sense of size and body log depth etcetera. But just that it allows it also allows some random input bits so some input bits are random and in addition to the actual input. So, there is random input and the actual input. So, that is r and c so it is a it is exactly like NC but there is scope for random inputs as well.

And although I did not mention it at that time so we saw that matrix multiplication is in NC and even determinant is actually in NC. So, fact determinant can be computed in NC 2. So, log n squared size depth is enough, so if the graph has a perfect matching, then the determinant of this quantity is non-zero and so the determinant of T is not 0 which means that if determinant is T is not 0 then which means that with high probability we will accept.

So, when will we accept, we will see we will accept whenever the polynomial entity testing algorithm will put some random values for x 1 1 x 1 2 etcetera and then evaluate the determinant of that that matrix. And this is very likely to be non-zero if the input is non-zero but there is a small chance that it becomes 0 so with high probability we will accept. If the graph does not have a perfect matching, then determinant T will be 0.

Which means whatever values you substitute it will still the determinant will still be 0. So, we will it is not high probability now we will always reject.

(Refer Slide Time: 44:13)



I will state the final result so let so perfect matching so I will just define the class or the language perfect matching or maybe I will call it bipartite perfect matching is a set of graph G, G is a bipartite graph G has a perfect matching. And what we showed now is that bipartite perfect matching is in RNC. Because a determinant computation is an NC and then we have to include accommodate randomness.

Because the algorithm that we have for polynomial identity testing is a randomized algorithm so this is in RNC. So, I will quickly summarize and then say some conclusions. So, the algorithm for randomized algorithm for polynomial identity testing was fairly straightforward just substituting some randomized algorithms. I stated the DeMello Lipton and Schwartz Zipper lemma which says that the fundamental theorem of algebra generalizes in the multivariate setting also.

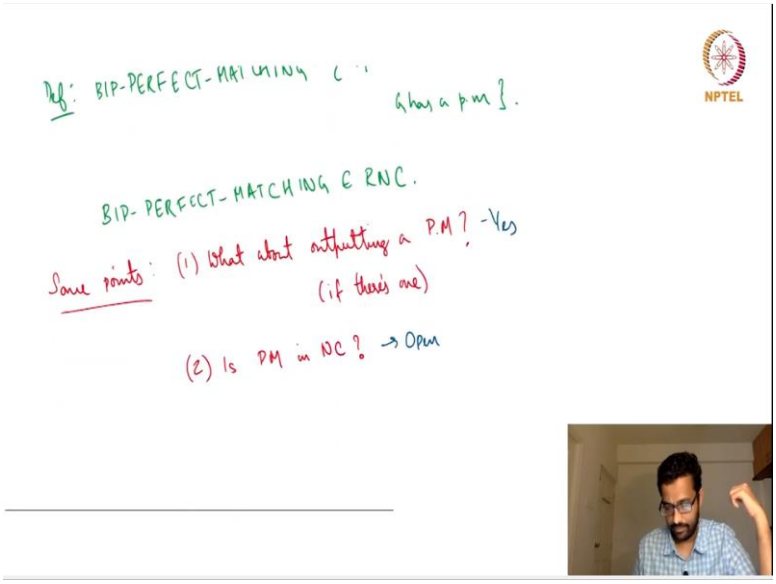
So, this is also an extremely interesting theorem lemma and has a very short proof so if you are interested you let me know I will share some references to this it is very simple very short. But just that we do not have time to get into the proof and I will be happy to share references if you are interested. Then we saw the problem of bipartite matching then we saw that graph has a perfect matching if and only if the permanent of the graph.

Permanent of the adjacency matrix is non-zero. But permanent is a quantity that is not so easy to compute. So, we went to the determinant but then that leads to errors so, instead of actually checking whether the determinant is 0 or not we check for the determinant of the of the cut

matrix where each value is replaced by variables. So, each one is replaced by variables 0 remains 0s and all we have to do is compute the determinant of this polynomial.

So, the matrix is has variables. So, the determinant will be polynomial compute the determinant of this matrix which will be a polynomial and check whether it is not identically 0 or not. So, we know that in this matrix graph has a perfect matching if and only if determinant is non-zero if this this is this equivalent is exact there is no error here. But checking for determinant equal to 0 identically 0 uses a polynomial identity testing. And that gives us that a randomized or RNC algorithm for the bipartite perfect matching.

(Refer Slide Time: 47:44)



Two points or maybe three points so one question; so this is a decision problem tells us whether answers whether the graph has a perfect matching or not. What about now outputting a perfect matching if there is 1, if the graph is a perfect matching can we output this perfect matching and the answer is that this is also can be done in the RNC perhaps we will see this in the next week's lectures.

And two is let me just say bipartite perfect matching or is perfect matching in general is this in NC. So, without use of randomization can we get a parallel algorithm for perfect matching. And it is still unknown. They are still open whereas the first one yes you can do it we will see it also this is still unknown there has been a lot of work even recently even the last 5 6 years there has been a lot of work where people have proved that for some special cases of graphs perfect matching is an NC.

For instance, and for planar graphs perfect matching is in NC but still the general question is open. So, the general question is still open on this and with that we come to the end of week 8.

So, we saw mainly we saw most of week 8 was taken by the proof that parity is not in AC 0 which was by the approximation we approximated each function or an AC 0 function we could approximate it using a low degree polynomial over F 3. And then we showed that parity cannot be approximated by a low degree polynomial over F 3. Then we saw the result that BPP has polynomial size circuits any language in BPP has polynomial say circuits.

Which; was just by noticing that every you can do the boosting enough number of times so that there is one random string that will be a witness for in all the input strings of a certain length. And if you hard code that you get a 1 1 circuit for all the inputs of a certain length. And without any randomness this is Adelman's theorem.

And then we saw today in this lecture we saw the polynomial identity testing and how it applies to bipartite matching. And finally seeing that bipartite perfect matching in bipartite graphs that it is in the randomized NC class and that is all for from me on week 8 and thank you.