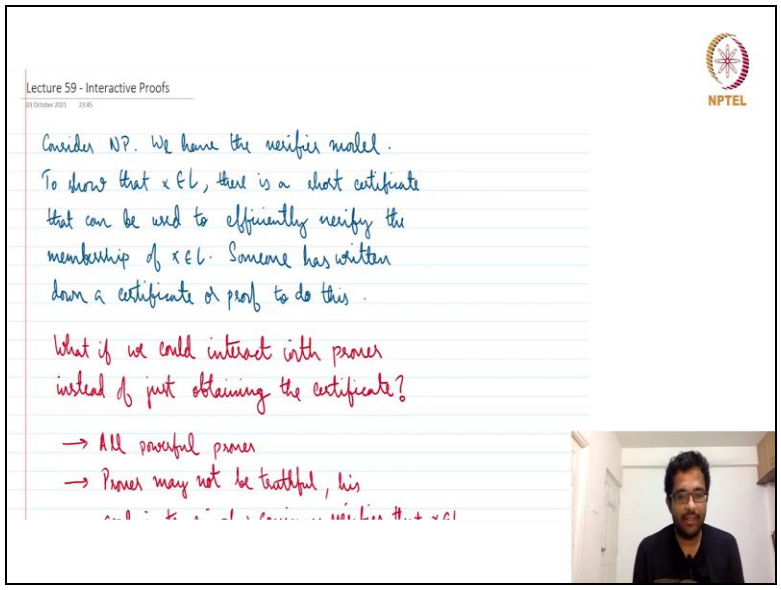
Computational Complexity
Prof. Subrahmanyam Kalyanasundaram
Department of Computer Science and Engineering
Indian Institute of Technology, Hyderabad

Lecture -59
Interactive Proofs

(Refer Slide Time: 00:15)



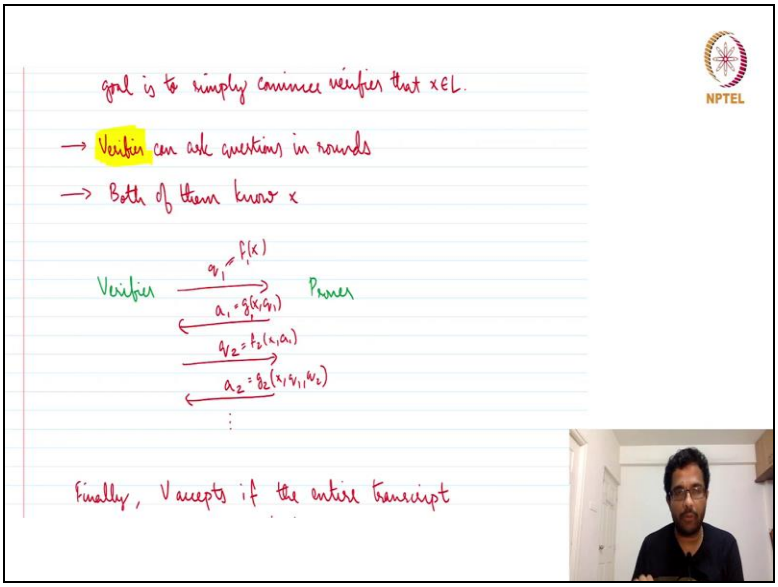
Hello and welcome to lecture 59 of the course computational complexity. This also be the beginning of week 12 and in this week we will see an exciting and interesting topic called interactive proofs. So, or in other words this will help us understand the power of interaction. So, to begin let us consider the very the verifier model for NP. So, we had this verifier model whenever x was in the language there had to be a proof or a witness or a certificate we use different terminologies.

Why such that a verifier machine the verifier is a deterministic polynomial time turing machine that could verify that x is in the language using the certificate. So, for instance when the language was 3 colourability the certificate could be the a proper 3 colouring. So, whenever the graph is 3 colourable there x is a proper 3 colouring that could be provided which can with which it can be verified that the graph is indeed 3 colourable.

If the graph was not 3 colourable whatever colourings you may provide one the it cannot be verified that the graph is 3 colourable because the graph is not 3 colourable. So, here we have this notion of somebody providing a certificate or a proof that the graph was 3 colourable or a given string x is in the language. So, one question that we could ask is now somebody is providing us this information what if it is.

So, now somebody is just giving us the certificate string. Now instead of giving it like that what if we had the capability to hold a conversation with that somebody who is providing us this information could we do something more. So, instead of just a deterministic polynomial time verifier that just verifies a given given x with a given proof y what if we could hold a conversation with the person who is providing the certificate. So, this is the main question that the area of interactive proofs ask.

(Refer Slide Time: 02:25)



What if we could have a conversation with the person who provides a proof. So, the person who provides a proof is called prover and the person who is actually sitting and verifying is called the verifier. So, let us let us try to see ah. So, let us we will slowly try to understand this and try to develop a model for this and then we will see various details and based on that various sub models of this of interactive proofs.

So, the model that we have is. So, in the NP we did not ask how we got the proof how the prover got the proof for the certificate if the graph was 3 colourable somehow magically the prover could give you a certificate which is a proper 3 colouring. If the formula was satisfiable he will just give an assignment that could be verified. So, the prover is considered to be all powerful and if something needed to be computed does not matter how much time it takes how much space it takes.

Whatever like he is not constrained by any resources he will just be able to execute the computation. The thing is the prover has this motive that or may have this motive that he wants you to convince to be convinced that x is in the language. So, you are you're trying to um. So, his goal is to by hook or crook to convince you that x is in the language. So, the problem there is that he may resort to undishonest methods to convince you that way. So, now the goal of the interactive proof systems is to develop ways where even we have to safeguard ourselves against dishonest provers.

But an honest prover and for an ex in the language should always be able to convince you that x is in the language. But when x is not in the language there could be dishonest provers who will try to convince you the text is in the language but the proof system has to be robust that he should not be able to convince you that way. So and they could hold a conversation. So, now how could that work.

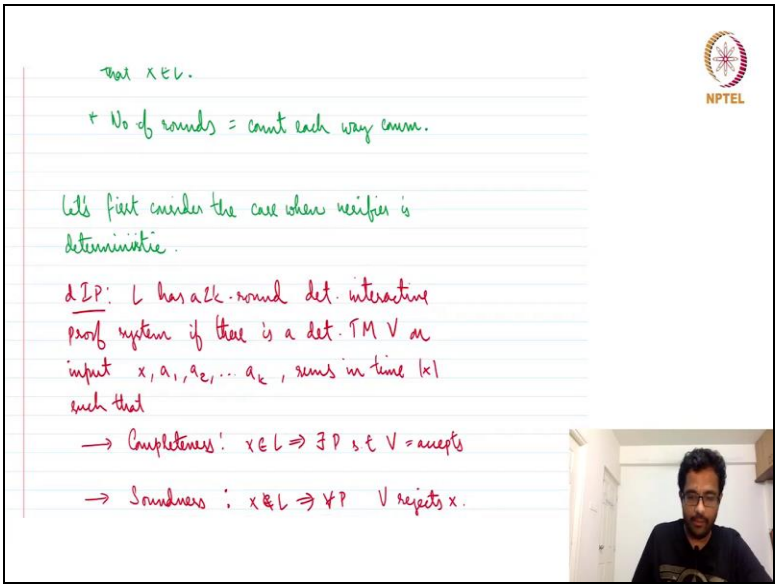
So, and the prover is all powerful and the verifier will be bounded. So, we will say the verifier can only do polynomial time computations. So, the model that I have drawn here. So, the verify of ask a question q 1 the prover replies a 1. So, both of them know the input x both prover and verifier know the input x they know that the goal is to convince the verifier that x is in the language but the idea of the protocol or the system that we designed should be that an honest prover should be able to easily convince the verifier that x is in the language.

But a dishonest prover when x is not in the language should not be able to convince the prover that x is in the language or if they try to do that it should be with limited success. So, both of them know x. So, maybe the verifier asks queries q 1 and to which prover responds a 1. So,

prover just is responding something based on what he knows which is x and the query that was asked and then prover again asks another question.

So, this question could be a function of x and the response that he received in the first query and then to that query maybe prover again responds with another a 2 which is another response and this may go on for a while.

(Refer Slide Time: 06:09)



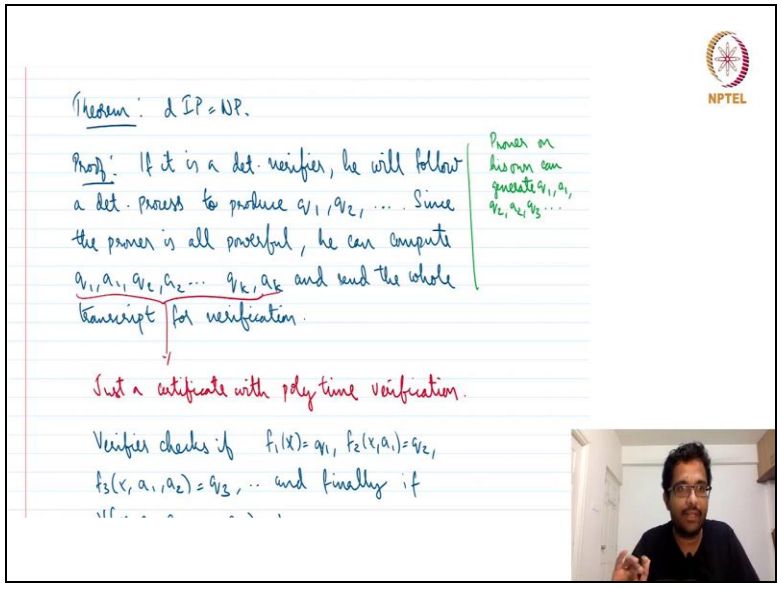
And finally some something like what we had in communication complexity. If the entire transcript is visible to both of them of course if the verifier can verify that access in the language using the entire transcript he did not use the entire thing but the entire transcript is there available for him to take made use of to be made use of. So, if the entire transcript can be used to convince the verifier that x is in the language he will accept.

So, some things again a bit more formally the verifier is polynomial time bounded he should compute everything in polynomial time. This also means that the prover cannot send huge messages or the messages the prover sends cannot be bigger than polynomially long polynomially meaning polynomial in the length of the input x because if he sends a long message that is super polynomial or exponential the prover does not have enough time to read the responses.

So, the transcript also has to be polynomial not just a single message even the entire transcript has to be polynomially wrong. Then the prover is all powerful he and as I said he need not be honest the goal for him is to simply show that x is not in l regardless of whether x was in l or x is not in l he will try to show that x is in l or he may try to show that x is in f. The system has to be has to be able to safeguard against such dishonest provers and the number of rounds is the number of times they communicate.

So, here q 1 is the first round a 1 the second round q 2 is the third round a 2 is the fourth round. So, like that I have seen in some places they count 2 ways communication is one round. So, q 1 a 1 together form one round but we will stick to the terminology where one way communication is one round. So, what I have marked in this figure q 1 a 1 q 2 q 2 a 2 together make four rounds sure.

(Refer Slide Time: 08:27)



So, to begin with let us consider the case when the verifier is deterministic. So, remember the goal the goal is to generalize NP where we just got a certificate and then we are verifying it. So, there is no interaction the prover just gives you a certificate and vanishes and now it is up to the verifier to do whatever he wants to do with a certificate. But now we are able to have a conversation. So let us say that the prover is deterministic.

So, it is called deterministic interactive proofs or DIP. So, we say that a language l has a 2k round 2k round deterministic interactive proof system if there is a deterministic verifier V. So, the verifier is just a polynomial time turing machine and in this case it is a deterministic turing machine that that just looks at the entire input. So, and basically it looks at x and all the responses that the prover gave a 1 a 2 etcetera I am not writing the q 1 q 2 because q 1 and q 2 were a functions of x, x and a 1 q 1 is a function of x q 2 is a function of x a 1 etcetera.

So, but then you could make the verifier recompute this function. So, the deterministic verifier just looks at all of this all of the responses and x and if x is in the language the prover should be able to come convince that x is in the language. So, that so, what we say is that there should be a prover that can convince the verifier to accept x. So, that there exists is a prover that can convince a verifier that x is in the language.

If x is not in the language no prover should be able to convince the verifier to accept x. So, for all provers verifier should reject x. So, it is kind of like the definition of NP but just that in NP we just had one certificate and that is it. So, if x was in the language there should be a certificate that leads to an acceptance there is one certificate. If x is not in the language whatever certificate you give it should not be we should not be able to verify that x is in the language.

So, it is kind of like that but instead of a static certificate we have the transcript of an interaction with the prover that is the only difference and somewhat it is not. So, difficult to see that this model where it is a deterministic interactive proof this model is no more powerful than NP why is that well um. So, if it is a deterministic verifier. So, the verifier keeps asking queries. So, if the prover. So, recall that the proverb is all powerful.

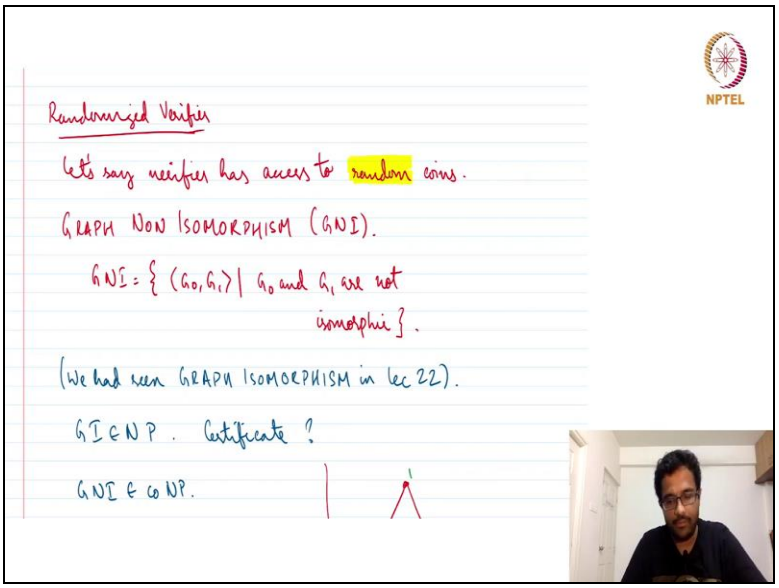
So, prover if we model the verifier to be a deterministic verifier. So, this means even the questions that he asks in each of these rounds all of this are part of the verifier. So, when the verifier is modern all of these queries are also deterministically generated. Now if the prover knows this the way the verifier is constructed the algorithms that go into the verifier and these are usually no secrets what these are usually not a secret.

So, in that case the proverb sees x and he instead of making the verifier ask the first query prover could himself ask the first query and output the answer or and figure out the answer that he would have answered anyway a 1. Then and because he can figure out the answer he can also ask the next question that the verifier would have asked because he knows the algorithm of the verifier and he can himself produce the next answer which is a 2.

Then he can himself the prover can himself ask q 3. So, the prover can himself generate q 1 q 2 q 1 a 1 q 2 a 2 everything. So, maybe just, so, prover on his own can generate q 1 a 1 q 2 a 2 q 3 and so on and at the end he will just send this entire transcript to the verifier look you would have asked me this then I ask you this then I will answer this then you would ask me that. So, the verifier just has to verify that yes correct I would have asked q 1 first the verifier can verify that then now if you give me a 1 as an answer then I would ask you to that again the verifier can verify because he would have generated this queries using a deterministic procedure.

So, he can verify that q a was q 1 followed by q 2 followed by q 3 and so on and finally he can also say that if you had given me all these responses a 1 to a k the verifier would have accepted. So, instead of having a conversation because verify the prover is all powerful the prover can himself provide the entire transcript for the verifier to just sit and verify in one shot. So, there is no need to have a conversation instead the prover just provides the entire transcript using his unbounded computational power and accepts if the entire.

(Refer Slide Time: 14:08)



So, this is what I said the verifier just checks if the first query is correct second query is correct and so on and finally if the transcript can be verified leading to x being in the language. So, if x is in the language there is an interaction which will lead to a certificate a transcript and if x is not in the language there is no interaction that will convince the verifier. So, there is no transcript. So, instead of having this conversation it could just be the proverb is giving this one short transcript.

So, in one shot he could give it. So, which is the same as NP where you just give a certificate and the verifier verifies it. So, same thing here, so, the verifier, so, deterministic interactive proofs is just the same as NP. So, we built up. So, much at the beginning about what if we could have a conversation with the prover and so on and somewhat disappointingly it does not really help if the deter the verifier is a deterministic turing machine.

So, let us see what else can we do if in this in the setup is it a useless thing. So, let us consider randomized verifiers. So, where the verifier has access to random coins and can generate can and execute random algorithms. So, all the queries that he generates could be a result of a random process the final verification could be result of a random process could be. So, all of this could use random coins. So, verifier has access to randomness.

So, just to given a an informal example which is also there in the Arura Barak. So, let us say that let us say that there are some socks. So, let us say red and blue coloured socks. So, and let us say I am colour blind which means I cannot tell them apart I cannot I have a red sock and blue sock and I cannot tell which one is which let us say I have red sock in my hand and blue sock in my left hand and let us see the prover is there but the prover knows is not colour blind and he can tell the difference between the red and blue.

So, proverb will say ok he will he will move away he will not look at me and what I can do to. So, but then he wants to convince me that these are 2 coloured socks because I am colour blind I cannot tell whether these 2 are different colours of the same colour but the prover says he can convince me that these are indeed 2 different colours. So, so I am colour blind I cannot really look at these socks and tell that they are the same. So, what we will do. So, this is an example of how randomness and interaction could together convince the colour blind person that these are 2 indeed different colours.

So, what we could do is I could just randomly decide to either keep the red at the and blue at the left or I could decide to swap them. So, I will randomly decide to do one of these and maybe I toss a coin if it is heads I will retain them tails I will swap them and then I will ask the verifier to approve her to come. So, prover knows the initial situation and then he sees then he can because he is not colour blind he will be able to tell oh yes you have swapped them or no you have not swapped them you have retained them.

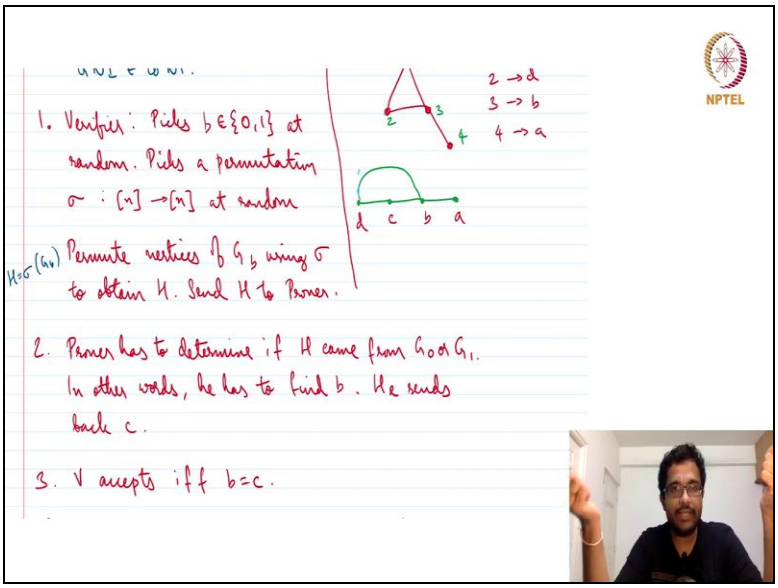
So, the first time we do this I will see that if there are 2 if they are indeed 2 different coloured socks he will be able to correctly say that I have swapped them or not but if I repeat it let us say 10 times and if he gets it all the 10 times then I will probably be convinced how can he be able to say correctly each of these 10 times. Whereas if they were both the same coloured socks let us say they were both blue then he would not be able to tell them apart regardless of whether i swapped them or not.

So, it is like a random thing so I will just swap them or not randomly. And he has to tell me if I have swapped but then there is no indicator for him to decide whether I have swapped them or

not both of them are blue. So, the best he can do is just simply guess something but whatever he does he will be with probability half because I may have or may not have swapped. So, the prover is powerful because I have the limitation of being coloured blind but the prover does not have that limitation.

So, that is the separation of powers that the prover and verifier have. So, this is an example of a protocol that the prover and verifier can together execute. So, the same thing we will exactly the same idea ok there is nothing more fancy but we will use it to address a problem called graph non-isomorphism.

(Refer Slide Time: 19:40)



In fact we have seen a similar or related problem called graph isomorphism in lecture 22 as a candidate problem which was NP intermediate I think we saw it in the context of Landers theorem. So, what are graph isomorphism's? Is isomorphism is 2 graphs. So, the problem in sense contains 2 graphs which are isomorphic. Graph of us non-isomorphism is a complement 2 graphs that are not isomorphic.

What do we mean by isomorphic it is just the same graph drawn differently or relabeled or whatever or if you would like to think in terms of matrices. Let us say take the adjacency matrix which you permute them in some way the rows in some way and columns in the same way. So,

you get a like it is like a relabeling of the vertices. So, just to give an example, so, here I have 2 graphs by the side 1, 2, 3, 4 is a graph and a, b, c, d are the vertices of another graph.

But if you look at them closely they are isomorphic. So, one may be mapped to c, 2 may be mapped to d. So, maybe I will just one may be mapped to maybe I will just write them by the side 1 to c, 2 to d, 3 to b, and 4 to a. So, you can see that here 3 and 4 are adjacent. So, b and a are adjacent c and b are adjacent. So, 1 and 3 are adjacent. So, you can you can verify that they are just the same graph relabeled and may be redrawn in a different way.

So, the question is if I give 2 graphs are they not isomorphic. So, because it is an estimate it is not really. So, we will look at graph non-isomorphism. So, are the given 2 graphs not isomorphic. So, these 2 graphs are isomorphic. So, we can do exactly the same thing that I said in the case of socks. So, before that what is it trivial algorithm for graph isomorphism trivial deterministic algorithm.

So, suppose they are both n vertex graphs. So, if one of them is n vertex and one of them is let us say n + 1 vertex graph then they are clearly not isomorphic because the number of vertices itself do not match. So, the issue arises only when they have the same number of vertices. So, suppose both of them have n vertices the one easy thing to do could be to try to generate all possible relabelling of one of the graphs and see if any of these relabel links match the other graph exactly but there are n vertices.

So, there are n factorial potentially n factorial mini relabellings for one of them and that is a is a is a lot of re-labeling to test. So, that is why it is in deterministic polynomial time but graph isomorphism is in NP because if you give me to isomorphic graphs I can give you a certificate. So, I can tell you the which relabeling maps G 0 to G 1. So, graph isomorphism is an NP consequently graph non-isomorphism is in co-NP but here we are looking. So, graph isomorphism is an NP.

So, clearly you can also do graph isomorphism in interactive proofs. So, a prover could just give the certificate but what is interesting is that graph non isomorphism we cannot seem to think of a

certificate that tells you that these are different. But there is a there is a proof system interactive proof system where the prover will be able to convince the verifier that the graphs are not isomorphic. So, we do the exact same thing in as we did in the case of socks.

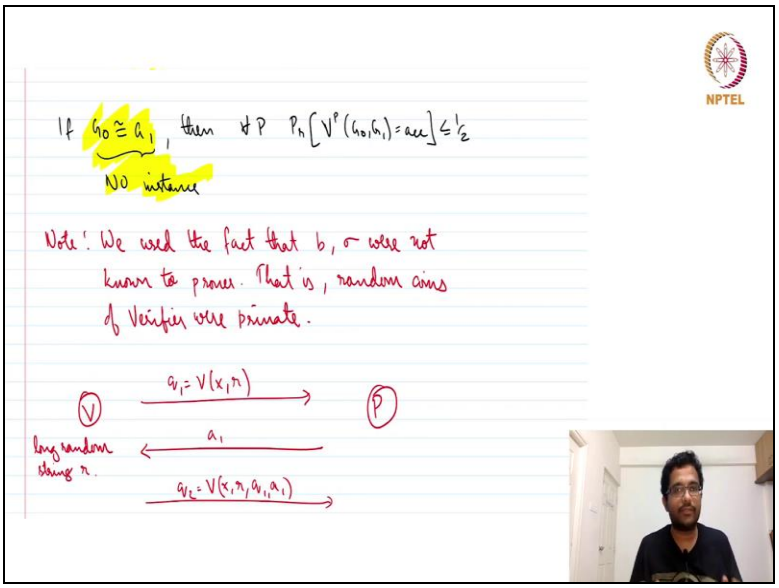
The verifier picks a random 0 1 bit and he picks a permutation which is a relabeling of the vertices and if he got 0 if b is 0 then he will permute G 0 with the with the permutation that he got if he got b equal to 1 then he will promote G 1 and the resulting graph of the permutation. So, the permutation the resulting graph is H notice that H is a result of both choices the random bit and the permutation random permutation because he permutes G 0 or G 1 based on which whether b equal to 0 or one based and the permutation is done by on the basis of the random permutation and the resulting graph is called H.

So, H may be just to be concrete H is equal to sigma of G b, H is sent to the prover and the proverb is all powerful. Now because he is all-powerful he is not computationally restricted he could look at H he could try out whatever he wants and if G 0 and G 1 are indeed different graphs he could try out various things with h and see which one it came from. But if G 0 and G 1 are indeed isomorphic graphs H could have come from either of them.

There is no way that the prover could tell where it came from because he does not know the random choice the b and the permutation. So, it could it could be a scrambled form of G 0 or scrambled form of G 1 because they are both the same thing if you relabel G 0 in some way you get G 1. So, you scramble G 0 in some way you scramble G 1 in some other way you could end up with the same graph.

So, there is nothing that the prover can do if they are the same graph except randomly get guess that b equal to 0 or something or b equal to 1, one of them. So, and the verifier will accept. So, verifier is expecting the prover to tell where the graph came from did h come from G 0 or G 1 if the verifier if the prover can correctly say this 0 or 1 each time then the prover the verifier will accept. If not he will reject.

(Refer Slide Time: 26:31)



So clearly if the graphs are not isomorphic as I said then verifier can tell you because he can just like the colour blind case he could distinguish in the in the non-isomorphic case he can distinguish. But if they are the same if they are the same graph if they are isomorphic so the notice that the non isomorphic case is is a yes instance for the language graph non-isomorphism and the isomorphic case is a no instance.

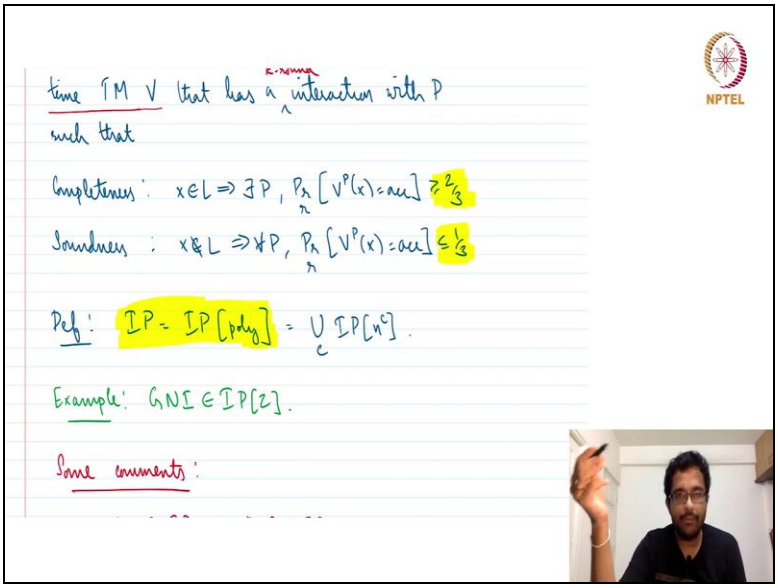
So, he cannot in the case of isomorphic case he cannot do better than half. So, if it is a yes instance he will make sure that the verifier accepts with probability one because he will always be able to correctly output the answer. If it is isomorphic he the best he can do is half. So, the best proverb whatever you try to do with the proverb the best one can do is half. So, there is a distinction here.

So, the probability that of acceptance by the verifier given whatever the prover could do if it is a yes instance there is a prover that can make him accept with probability one if it's a no instance whatever the prover does whichever approval it is the best he can do is half. So, there is a gap between the essence and no instance and notice that here we use the fact that the choice of b the random bit and the choice in the permutation were both withheld to the prover they were not provided to the prover.

So, the random choices made by the verifier were private they were not told to the prover. So, note this is very important aspect of this proof if either b or sigma was revealed to the prover then the prover could immediately tell where it came from. So, this is an example of an interactive proof 2 for graph non-isomorphism and clearly as I said the graph isomorphism is already has an interactive proof.

So, this is an example of a language which is not known to be an NP, G n graph non-isomorphism is in co-NP and we do not know if it is an NP yet but we have an interactive group. So, this already indicates that the class of languages that have interactive crew systems seems to be more powerful than NP because we do not have it we do not know whether graph non-isomorphism is NP.

(Refer Slide Time: 29:12)



So, just to formalize the setup of interactive proof systems again we have verifier and prover the the prover. So, the verifier poses quest queries 1 q 1 q 2 q 3 and the verifier provides sorry prover provides responses a 1, a 2, a 3. So, q 1 is the verifiers sorry verifies first query using x and a random string the random input. So, the random input you could think of it as maybe a long random input and prove a verifier using different chunks of it each round some response comes back a 1, q 2 is depends on the input x the random string r the first query q 1 and the response first response a1 and so on.

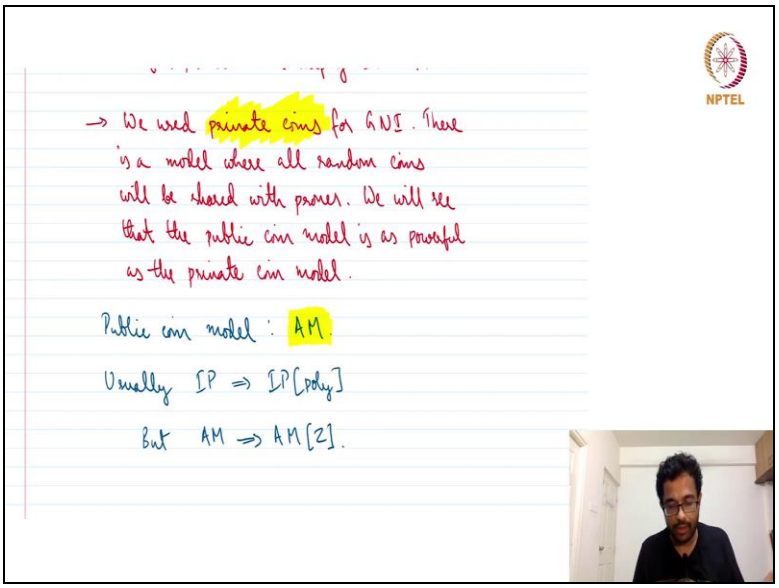
And at the end based on the entire transcript it is decided to accept or reject. So, we say that a language L is in IP k. So, what is IP k? IP k is the interactive proofs with of k rounds. If there is a probabilistic polynomial time machine V, so, we are not going to talk about deterministic verifiers from. Now once everything is going to be probabilistic or randomized verifiers V that has a K round interact interaction with approver P such that if x is in the language there is a prover that can convince the verifier to accept with probability at least 2 thirds.

And if x is not in the language any prover that you that; you can think of can convince a verifier to accept with probability at most one third. So, the first part is called completeness and the second part is called soundness. So, completeness corresponds to when x is in the language soundness corresponds to when x is not in the language. So, there is a probabilistic polynomial time verifier we that that has a k round interaction with approver such that when x is in the language the prover makes the verifier accepts.

There is a prover that can make the verifier accept with probability at least 2 thirds and when x is in the not in the language. So, when x is in the language there is a prover that can convince a verifier to accept the probability at least 2 thirds and when x is not in the language any prover the best they can do is to convince the verifier with the probability at most one thirds to accept the string x.

And we will when we say IP without so I said L is equal to IP k for a k round interaction when we simply say IP without a cave associated with it this indicates IP of polynomial number of rounds. So, they can they can have potentially polynomial mini rounds of communication and what we have seen. So, far is that graph non isomorphism is in IP 2 because there was 2 rounds of communication the verifier conveying the random bit the not the random bit the permuted graph H and the prover conveying the purported random bit or the random bit that was that he thinks was used.

(Refer Slide Time: 32:57)



So, 2 rounds some comments so far some remarks. So, we will close this lecture with these remarks. So, first of all this was a generalization of NP which should not be that difficult to see because instead of verifier just or the provers is giving a certificate. Now there is an interaction going on. So, and it is we saw deterministic IP is equal to NP randomize has only has more power. So, there is something that the NP is contained in IP because a language in NP the prover could just give the certificate and make the verifier verify that is one proof system.

Another thing BPP is also an IP because the verifier himself has the ability to use random coins. So, he could just disregard the prover and do everything himself and this will be a BPP machine. So, if the verifier just ignores the prover he is like operating as a BPP machine if the verifier just ignores randomness it is like deterministic interactive proof which is NP. So, both NP and BPP are sub classes of IP interactive proofs.

And another way to say it is that interactive proof generalizes both of these and like in BPP we could improve the probability. So, that nothing is special about 2-thirds and one-thirds we could improve this probability to very close to one and very close to 0. So, one example is one one like one minus one divided by 2 power m and one divided by 2 power. So, it could make them exponentially close to 1 and 0 respectively by exactly the same argument as BPP boosting we can do multiple rounds of this.

So, you do one round you do 2 rounds you do 3 rounds and take the majority whichever one it comes. Exactly the same arguments bound everything goes through one small point which I will state but which actually requires a very sophisticated proof is that whatever I just described is a CDL process and this can be made into a parallel process also. Now 2 points or one point rather.

So, we said that the verifier when we moved from a deterministic verifier to a randomized verifier the power the capability is increased. So, deterministic verifier was NP but randomized verifier seems to be able to do more things what if we made the prover also randomized or what if the proverb was able to do probabilistic stuff well this does not really help because as it is the prover is considered all powerful.

So, as we know BPP is contained in sigma 2 for instance. So, there is a which means there is a higher class of non-randomized machine that can do whatever BPP does. So, there is nothing much to be gained or in other words if there are multiple options available to him and if he if the proverb is executing a random protocol then because he is all powerful he can he can also see the best best possible out of the many things that he does, so, he can always choose the best option best random coin.

So, there is no benefit gained by making the prover randomized another point is that the entire transcript is a polynomially many rounds and then each round has polynomially wrong transcript. So, entire transcript is a polynomial sized. And and the verifier can just verify that this is this corresponds to something a proper proof or not. So, suppose there was no proverb suppose the verifier had to do the work of the prover.

So, in other words what is the how powerful can the proverb be beyond which it doesn't really help the answer is we can replace the prover by a P space bounded machine the space machine because. So, the prover can see what is the best way to convince the verifier by just going through all the possible transcripts? In other words another way to look at it is that if a language is accepted because of this interaction.

We can just another way to decide the same language is to just try out all the possible transcripts and that all the possible transcripts is of polynomial size. So, I can try out all the possible polynomially long transcripts and check that this is a valid transcript and then accept or not accept x based on all of this and this process requires only polynomial space because the length the size of the transcript is polynomial.

So, the entire process this entire if a language is has an interactive proof this language also has a can also be decided in P space. So, this is one another point now 2 further points that I would like to state one is perfect completeness which means. So, recall that completeness is the case when x is in the language. So, we said that the probability of acceptance of x should be at least 2-thirds can we and we already saw that 2-thirds can be pushed very close to one.

Can we push it to exactly one and the answer turns out to be yes using some ideas similar to the proof that BPP was in sigma 2 and appropriately modifying the verifier we can make perfect completeness meaning the completeness part here this probability 2 thirds this can be pushed to one. So, perfect completeness. So, any interactive proof you can replace with an interactive proof of perfect completeness and the next thing is perfect soundness.

So, soundness is the situation where x is not in language. So, can we push the probability of acceptance when x is not in the language from one thirds down to 0 we already saw that we could make it exponentially close to 0, 1 by 2 power something can we push it to 0 and this turns out to be this looks unlikely we do not know how to do that because if we have perfect soundness and perfect completeness this is like a deterministic interactive proof system which we already know is equal to NP.

So, unless the entire class of IP is equal to NP perfect soundness will not be possible finally one more point we used private coins for the protocol where for graph non isomorphism meaning it was important that the random bit b and the permutation sigma were withheld from the brewery that protocol would not have worked if this had to be made public this had to be told to the brewer. So, can we have interactive proofs where the random coins are public.

So, this one may think that this weakens the entire system because the key power in the graph nano isomorphism protocol came from the fact that some information was hidden from the approver the random coins the prover is all powerful but he does not know the random coins because he can do any computation but there is some input the random coin that he did not know. So, if when we move when we insist that the random coin should be made public we may think that the prover will lose some power or the proof system may rule may not be able to do some things.

So, interestingly it can be shown that whenever there is a private coin protocol you can get a public coin protocol sometimes with a bit more rounds but whenever there is a private coin protocol you can also get a public growing protocol. So, protocols with public coin proof system are as powerful as those with private coin proof system. So, this is a slightly interesting somewhat surprising fact and some small notations.

And before I conclude this lecture the private coin setup is called IP as we defined already and the public coin model is called am it is short for Arthur Merlin. So, I will explain when we talk about AM and there is a one small interesting fact is that when I say IP without any number of rounds specified the implicit the what I want to say is that it stands for ip in with polynomially many rounds but when one says M what they imply what they want to say is aim with 2 rounds through our send something verifies in something 2 rounds.

So, this is a bit of an inconsistency with the notation that but then that seems to be the standard thing that people are following inconsistent notation but unfortunately this seems to be the standard. So, we will have to just get used to this and uh. So, in the coming few lectures we will see IP what how powerful IP is we already saw that IP can do some things that possibly are not there in NP we already saw that IP is contained in P space.

So, it is not more powerful than P space but how far powerful it is we will actually see that IP is as powerful as P space or I will at least mention that and then we will see different aspects like how different aspects with about in public coin and private coin protocols how they relate to each other and how sometimes interactive proof systems can have also been used to infer something

about other aspects of complexity we will derive at this one statement which has nothing to do with interactive proofs but proved using the interactive proof systems and yeah that.

So, that in this lecture we just defined interactive proof system we formally defined it and we saw deterministic interactive proofs. The interactive proofs which were the verifier can be randomized and then we saw a bunch of remarks and that's all i want to say, thank you.