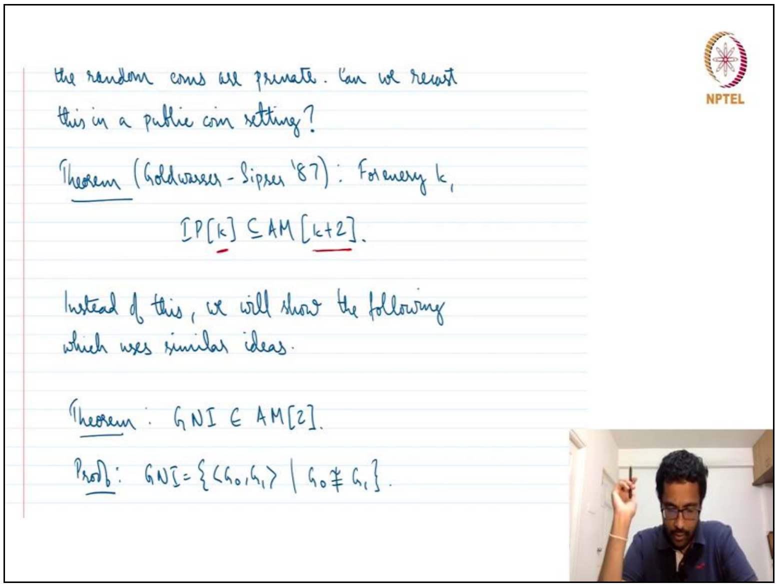
Computational Complexity
Prof. Subrahmanyam Kalyanasundaram
Department of Computer Science and Engineering
Indian Institute of Technology, Hyderabad

Lecture -62
Simulating Private Coins using Public Coins

(Refer Slide Time: 00:15)



the random coins are private. Can we recast this in a public coin setting?

Theorem (Goldwasser - Sipser '87): For every k,

$$IP[k] \subseteq AM[k+2].$$

Instead of this, we will show the following which uses similar ideas.

Theorem: $GNI \in AM[2]$.

Proof: $GNI = \{\langle G_0, G_1\rangle \mid G_0 \not\equiv G_1\}$.

Hello and welcome to lecture 62 of the course computational complexity. So, we saw interactive proofs with. So, we first initially defined then the define the class IP which said just said that there should be communication there should be interaction between the prover and verifier. Then we saw the classes Arthur Merlin classes where we said that the verifier has to compulsorily send the random bits across to the approver.

And so, having to compulsorily send the random bits to the prover is is kind of a restriction because we saw that in the case of graph non-isomorphism proof we could benefit by hiding the random coin from the prover. So, the question is. So, this restriction how does it make things how restrictive is this restriction is there something that we cannot do with insisting on public coins that we could have done with private coins.

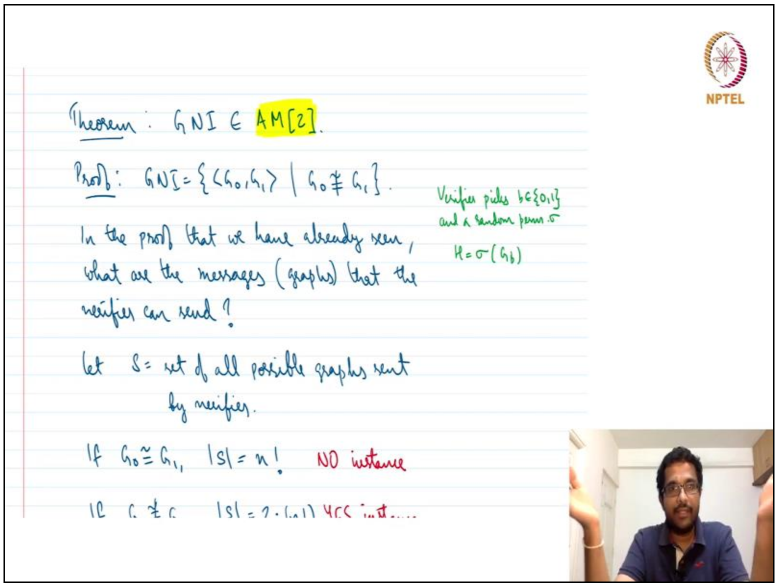
Was there some language that we could decide using an interactive proof system with private coins but cannot decide with the public coins system. So, that is one question. For instance is can you can you decide graph non-isomorphism in a public coin setting how many rounds does it take what resources does it take. So, this is a question. So, this was this was answered by Goldwarres And Sipser who said that for any k any any knee not be a constant if something can be accomplished by an interactive proof in k rounds then the same thing can be accomplished by a public coin interactive proof in k plus 2 rounds.

So, basically 2 additional rounds are enough to convert a private coin protocol to a public coin protocol. So, at first this seems kind of impossible because in graph non-isomorphism the keeping the random coins private was essential. Now how is it possible to convert it into a public coin protocol where the verifier has to send across the randomness. Well we already saw some example in the in the three in the sharp 3 SAT instance where we could use randomness in a in a clever way.

So, the key thing there was that we will we will the interact the verifier will generate random coins only when required and the prover was forced to commit certain things before knowing the randomness. And because he did not know what randomness is going to be generated later because well they are random that that helped the verifier that that helped the prover or that helped the system to be more robust.

So, that that prevented the very the prover from cheating because he does not know what randomness is going to show nor does the verifier no but verified this does not care verify just is waiting for the prover to convince him. So, the idea is that we will send information only when or we will send the randomness or generate the randomness only when necessary okay.

(Refer Slide Time: 03:43)



Theorem: $GNI \in AM[2]$.

Proof: $GNI = \{ \langle G_0, G_1 \rangle \mid G_0 \not\cong G_1 \}$.

In the proof that we have already seen, what are the messages (graphs) that the verifier can send?

Let S = set of all possible graphs sent by verifier.

If $G_0 \cong G_1$, $|S| = n!$ NO instance

If $G_0 \not\cong G_1$ $|S| = 2 \cdot (n!)$ YES instance

Verifier picks $b \in \{0,1\}$ and a random perm. σ

$H = \sigma(G_b)$

So, this was a Goldwasser and Sipser's result. So, in instead of showing this result in the main in the full general setting we will in fact we will just show that graph non-isomorphism can be this can has a public coin protocol with just 2 rounds. So, in fact graph non-isomorphism has a Arthur Merlin protocol with 2 rounds.

So, we actually we had already seen that graph non-isomorphism has an IP 2 now we say that it is an AM2 as well. And the ideas are very similar. So, we will it is it is probably more instructive or simpler to see the in the in the for a specific problem for a specific case rather than the general situation. So, the proof will be easy to follow once you understand this. So, the result that we will see again using Goldwasser this is also Goldwasser and Sipser is the graph non-isomorphism has an AM protocol.

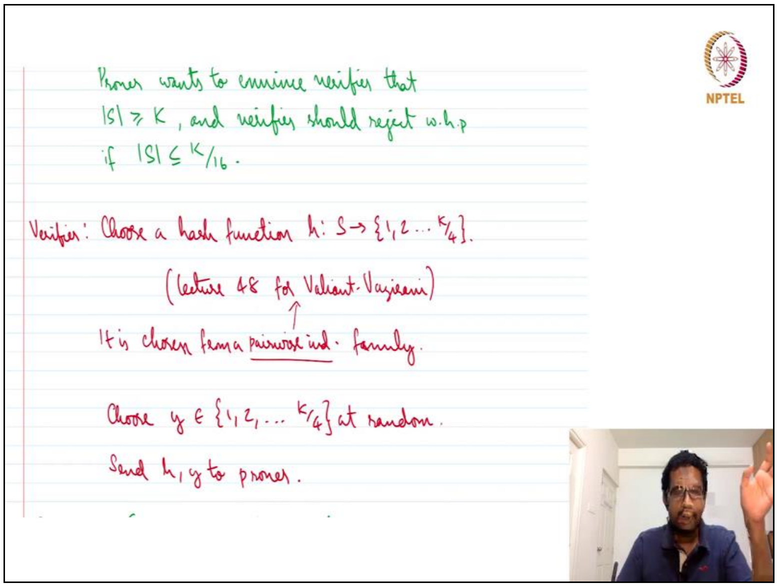
A improved system with 2 rounds ok. So, let us recall what was graphed non-isomorphism. It consists of 2 graphs such that they are not isomorphic G 0 and G 1 and we saw graph nonisomorphism in the previous lectures. And so, what was the earlier protocol? The earlier protocol was that verifier picks b from 01 and a random permutation sigma and then sends H which was sigma of G b.

So, if b was zero he would send a permutation of G 0 if b was 1 then he would send a permutation of G 1 and the prover had to decide where this graph H came from did H come from

G 0 or G 1. If G 0 and G 1 are isomorphic then h could have well come from either one there is no way the prover could tell. If the graphs are not isomorphic then the prover using his infinite power should be able to compute and figure out where it came from.

And that was the principle that we used for the private coin protocol private coin proof system for graph non isomorphism. So, the message that the verifier sends is the is a graph H. So, what are the possible graphs that the verifier could have sent?

(Refer Slide Time: 06:28)



Prover wants to convince verifier that $|S| \geq K$, and verifier should reject w.h.p if $|S| \leq K/16$.

Verifier: Choose a hash function $h: S \rightarrow \{1, 2 \dots K/4\}$.

(lecture 48 for Valiant-Vazirani)

$\uparrow$

It is chosen from a <u>pairwise ind.</u> family.

Choose $y \in \{1, 2, \dots K/4\}$ at random.

Send h, y to prover.

Let s be the set of possible graphs that the verifier could have sent now if G 0 is equal to G 1 which means it is and no instance for G 0 is isomorphic to G 1 which means it is in a the graphs are isomorphic. So, it is a no instance for the graph non-isomorphism problem because it is a graph non-isomorphism is a language. So, it is a no instance for the graph non-isomorphism problem in that case G 0 and G 1 are the same graph.

So, the set of graphs that could be sent by the verifier is just the permutations of that one graph there are n factorial ways to permute a graph or permute the vertices permutation is just a reordering or relabeling of the vertices. So, there are n factorial different graphs that the verifier could send well there is a small detail here on something called automorphisms. So, we are currently ignoring it. So, automorphisms are basically some relay building that what if multiple relabellings look the same.

So, for instance some simple graphs if you relabel it in 2 ways it still look the lengths look the same. So, it will actually bring down the size of s to something below n factorial but I do not want to get into that mainly because that particular detail is not really that crucial for this proof. So, I will tell you why it is not crucial. If G 0 is not isomorphic to G 1 which is which means G 0 and G 1 are not isomorphic which is a yes instance for graph nouisomorphism then the set of graphs that can be sent by the verifier there could be n factorial permutations of G 0 and n factorial permutations of G 1.

So, there will be 2 times n factorial graphs that the verifier could possibly send. Again there could be automorphisms but that will again bring it down from 2 to n 2 times n factorial to a smaller number but the key thing is that what we will exploit in this is the number the size of s which is a set of possible graphs that the verifier will send the size of s how different is it in the yes case and the nose case.

So, we want a sizeable gap between these 2 between the number of graphs that could be sent to the yes case and the number of graphs that could be sent in the no case. So, the exact number n factorial is not that crucial what we are interested is in the gap and even the automorphism can be fixed by doing something. So, you can you can look at Arora Barak where they do something to fix this automorphism issue.

But I do not want to get into that detail because it is just because it is not very instructive to on how to fix that you can read it if you are interested. I want to just touch on the the core idea that is used by Goldwasser and Sipser and what they use is the way to tell the differences of the sizes in the essentials and the no instance and just to amplify. So, now as it is there is a between the s unknown since there is a factor of 2 difference 2n factorial and n factorial just to amplify it a bit further let us consider the set s prime okay s prime is just a Cartesian product of s.

So, think of it as instead of Cartesian product you can think of the verifier generating 4 random bits and 4 permutations and sending 4 graphs. So, h 1 which is a permutation of one of them h 2 which is the permutation of another one h 3 is the perpetration of another one and so on. So, what is the total size of the of the set that verifier could have sent it from. If it is a yes instance, so now you can think of it as a the whole the thing the set of things that that is sent by the verifier is from s cross s cross s cross s.

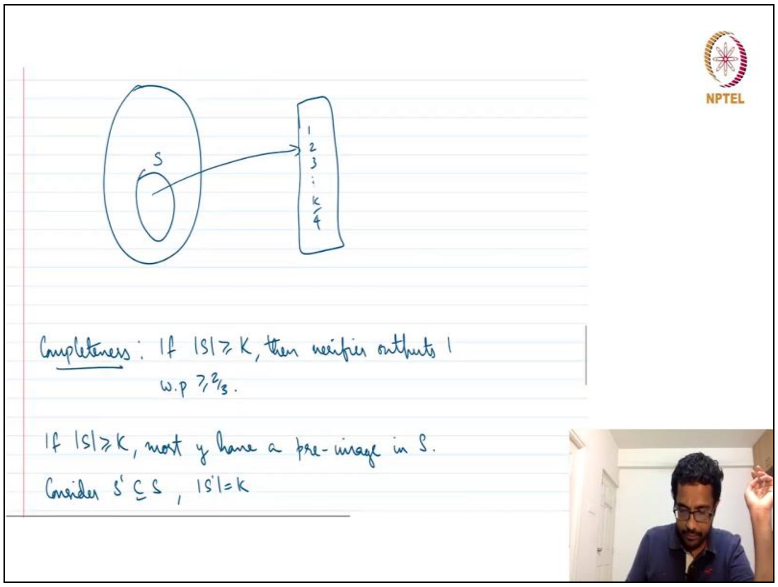
So, it is n factorial power 4 if it is a no instance because both the graphs are the same and 2 times n factorial power 4 which is 16 times n factorial power 4. So, when it is a yes instance. So, there is a no instance and this yes instance. So basically it is just a fourth power of the previous answers. So, here the thing is that because by taking the fourth power we have amplified the gap. So, here we have amplified the gap from 2 to 60 this is a bit convenient even with 2 I think you can reason a proof but then a bigger gap is makes for an easier explanation.

So, and the protocol that we will see is called Goldwasser Sipser set lower bound protocol that helps us distinguishes distinguish between these 2 sizes in factorial power 4 and 16 times n factorial power 4.. So, in factorial power 4 is not very important what is important is the 16 factor gap okay. So, all that we will need is the 16 factor gap. So, it is like this there are 2 possibilities that the verifier is in or the to the 2 possibilities that they are in either the graphs are isomorphic or they are not isomorphic.

So, in one case the number of number of messages number of messages that can be sent by the verifier is k on in the other case the number of messages that can be sent is 16 times k or maybe in another word in other words it is like in one situation the number of messages is k and the other situation it is k divided by 16 the 16 factor gap and the protocol will try to tell if there is a there is a difference between these numbers it will tell whether you are in the k case or k divided by 16 k’s.

So, let us see and the next proto next property that we will use is that we will come to that I will explain when we come to there. So, choose a hash function h that maps s to the set 1 to 3 up to k divided by 4. So, notice that k divided by 4 is has a 4 factor away from k and 4 factor away from k divided by 16. So, it is somewhere in the middle and this is chosen from a pairwise independent hash family. So, this we had seen in lecture 48 where we explained Valient and Vazirani lemma.

(Refer Slide Time: 13:30)



Completeness: If $|S| \geq K$, then verifier outputs 1 w.p $\geq 2/3$.

If $|S| \geq K$, most y have a pre-image in S.

Consider $S' \subseteq S$, $|S'| = K$

So, pairwise independent hash family is basically a family of functions that behave in a random like manner in the formal definition you can go back and refer to lecture 48. So, if there are 2 2 values 2 hash functions they will they will act as if they are independent with each other. So, the probability that h of x maps to let us say 1 will be h of x will be could be either one of one 2 three up to k by four with e all with equal likelihood one divided by 1 by k 1 divided by k by 4.

So, it behaves in a random like fashion that is all that we require for this proof. So, if we choose a verifier chooses a hash function from this pairwise independent hash family okay and then it chooses a y from the set 1, 2, 3 up to k by 4. So, basically from the the range of the hash function and the verifier sends h and y to the prover. So, notice that the verifier only chooses the hash function and the the number y both of which the random choices are just being sent to the approver.

So, the it is sending the entire randomness again this is what we want in a public coin protocol and what does a prover do the prover finds and the provers does the prover is all powerful the prover has to find some element x in s such that x maps to y. So, it is like this. So, this is this is the set 1, 2, 3 up to k by 4 and this is the set s. Now given a certain y the prover has to find out is there an x that maps to that y and and if there is an x that maps the y the prover will send that value x to the verifier.

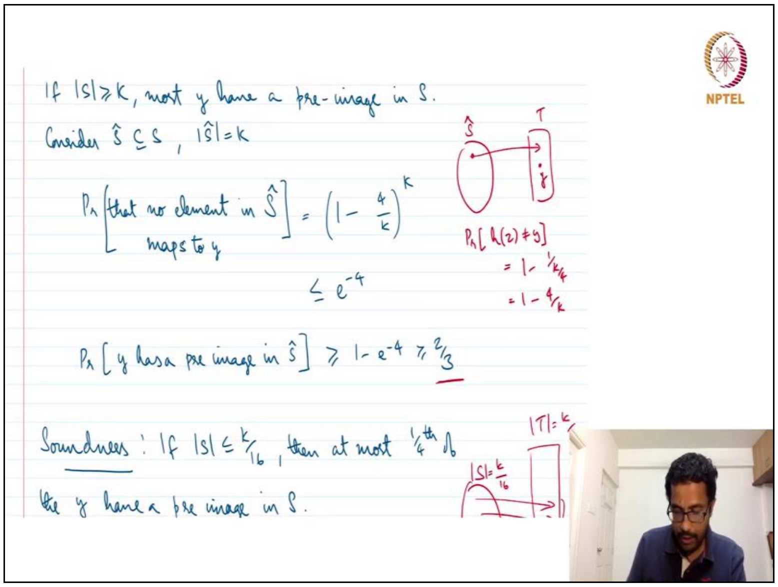
So, notice that the verifier cannot identify such an x because s itself is like n factorial or some big number. So, it does not have the time to go through all the elements of x s. So, but prover can send the find and send x and notice that so s x is an s means x is possibly an element that could have been sent by the x is not possibly an element that could have been sent by the verifier and the prover can say that well this is an element that you could have sent because you take G 0 and you permute it you get x or you take G 1 and you permute it and you get x this is something that the prover can tell the verifier.

So, it can tell the proof that or the certificate that x is in s because s itself is too big to list down but to convince the verifier that x is an s it can tell that take. So, this is just an example certificate will be like take G 0 and permute with let us say sigma 1 to get x something and this the verifier can verify. So, so that the prover does not cheat him saying that this is something. So, now given x the verifier can verify that x is indeed an element of s and second the verifier can also verify that x maps to y because the hash function is easy to compute.

So, once this is done one of the verifier verifies this that x is in s and h x is equal to y it accepts verifier access if he cannot verify this he will reject. So, the prover cannot lie because the verifier is verifying everything. So, the only thing the; so, the prover will be successful if he can find a y he can find a find an x in s that is a pre image of y. So, prover's goal is to find an x that is a pre image of y in s. So, this is the goal and when will the very approver be able to do that and all the probability completeness and soundness boils down to this how likely is the prover able to find a pre image of y in s that is it.

So, this is s this is the set 1 2 3 up to k by 4. Now how likely is the prover find able to find a pre image of y.

(Refer Slide Time: 19:03)



If $|S| \geq k$, most y have a pre-image in S.

Consider $\hat{S} \subseteq S$, $|\hat{S}| = k$

$$\Pr\left[\begin{array}{l}\text{that no element in } \hat{S} \\ \text{maps to } y\end{array}\right] = \left(1 - \frac{4}{k}\right)^{k}$$

$$\leq e^{-4}$$

$$\Pr\left[y \text{ has a pre image in } \hat{S}\right] \geq 1 - e^{-4} \geq \underline{2/3}$$

Soundness: If $|S| \leq k/16$, then at most $1/4$th of the y have a pre image in S.

So, let us say the completeness means it is a yes instance which means the graphs are non isomorphic in which case the set s is large. So, completeness means it is non isomorphic. So, it is s is of size at least k then what is the probability that the verifier will be convinced if s is of size k what is the then the point is that most y will have a pre image in x s because s is of size k. So, you map each element of s into let me call this set let me call this at T. So, T is equal to 1 2 3 up to k by 4.

So, when s is as big as k then most elements within T will be mapped by some element in s because simply because s is 4 times bigger than t. So, that is the intuition. So, consider so, to see the probability consider s hat which is a subset of s. So, suppose s is at least k. So, consider s hat as a subset of s where s hat is of size exactly k. Consider s hat and of size exactly k now what is the probability that no element from s hat maps to y.

So, s hat is of size exactly k. So, is exactly k and here we have t and there is a certain y. So, what is the probability that an arbitrary element misses y what is the probability that an arbitrary element misses y the probability that an arbitrary there is a it is a pairwise independent hash family. So, an arbitrary element misses y let us say with probability. So, one element misses y. So, probability that let us say z, h of z for a specific element z is not equal to y it is equal to.

So, probably that h of that equal to y is 1 divided by k by 4. So, this is 1 - 1 divided by k by 4. So, in other words it is 1 - 4 by k that z does not get mapped to y what is the probability that none of the elements in s hat maps to y it is 1 - 4 by k whole power k because there are k elements in s hat and this is easily upper bounded by e power -4 this is easily upper bounded by e power -4. So, this is the probability that no element in s hat maps to y.

So, what is the probability that y has a pre image that is the complement event some element maps to y. So, what is the probability that y has a pre image it is at least 1 - e power -4 which is way bigger than 2 thirds this is e 1 - e power -4 is much closer to 1. So probability that y has a pre image in s hat is at least 2 thirds in the yes instance. And in the no instance we want to show the probability that y has a pre-image is at most one-fourth.

So, that is the soundness suppose the size of s is at most k by 16 this is s and this is T. So, let me we just write T S bigger or draw T as bigger then s has at most k by 16 elements and T has k by 4 elements. So, size of T is k by 4 size of s is k by 16. So which means even if; each element of s maps to a distinct element in T even if that happens at most one fourth of the elements of T are covered as images of elements of s.

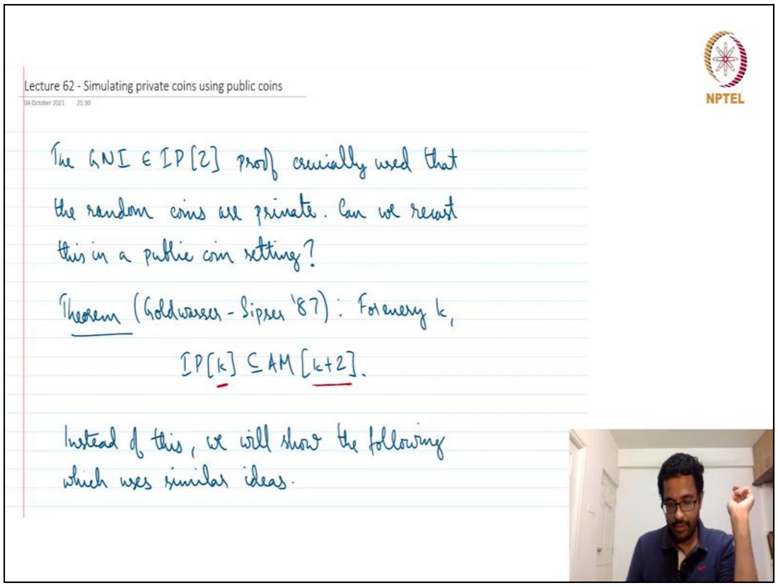
So, at most one fourth of the elements have images in s. So, which means at least three fourths of the element in s do not have pre image in s. So, the probability that y has a pre image in s is less than one fourth. So, it has to be. So, the the size of the number of elements in t that have pre image and s will be at most one false T it cannot be bigger than this. So, the probability that y has a pre majorness is at most one-fourths.

So, it is it is one-fourth which is less than one-thirds. So, the completeness and soundness have been checked just one more point in the completeness we said that the probability y has a pre-image in s hat to be at least 2 thirds we said that s is super set of s hat. So, the probability that y has a pre image in s will be at least this. So, maybe I will just write that here probability that y has a pre image in s is less than or equal to this.

So, we have completed the completeness and soundness and using this protocol the prover and verifier can be convinced or the verifier can be convinced by the prover that either the graphs are non isomorphic or they are isomorphic or they are not non-isomorphic. So, notice that once again it is a public coin protocol because the verifier sends h and y which were the random choices and the probabilities check out.

And it is a 2 round protocol the verifier since h and y prover sends a certificate and then the verifier the verifier does not do any other random generation or anything he just performs a verification and that is it.

(Refer Slide Time: 25:25)



Lecture 62 - Simulating private coins using public coins

04 October 2021 21:30

The GNI ∈ IP[2] proof crucially used that the random coins are private. Can we recast this in a public coin setting?

Theorem (Goldwasser - Sipser '87): For every k,

$$IP[\underline{k}] \subseteq AM[\underline{k+2}].$$

Instead of this, we will show the following which uses similar ideas.

So, this completes the proof that graph non isomorphism has a 2 round Arthur Merlin proof Arthur Merlin in a sense public coin interactive proof for graph non-isomorphism and the proof that for the general Goldwasser and Sipser theorem is also very similar but I am not getting into that. So, there we require 2 additional thumbs but but when k is constant AMk + 2 also is a constant. So, we are we have already seen that AMk + 2 is equal to am when k is a constant because k + 2 is also a constant.

But in other words another point is that when k is polynomial then when you have polynomially many rounds IP and am are the same because you just have 2 more rounds it is still polynomial. So, when you have polynomial many rounds public coin private coin does not really make a difference because we are both polynomial anyway. So, we have seen the proof that the Goldwasser and Sipser proof that you can convert private coin protocols into public coin protocols.

The key idea that we used was the difference in the sizes of the sets. And that completes the main theorem. And that completes most of almost all of what I had to say on interactive proofs. So, let me just summarize what we have seen. So, far we saw interactive proofs which were which generalized this NP and BPP where you could have a interaction with the prover not just operate on a stat a static proof given to you and how powerful is this.

So, we first saw that when the verified is deterministic it is only as powerful as NP. So, then we moved to a randomized verifier then we saw properties and then we saw this graph noun isomorphism proof with which is a problem in co-NP using a private coin setting then we saw properties like completeness soundness and so, on. We said that ip is contained in P space and we saw the proof that sharp 3 SAT is contained in IP.

So, which is an actually a sharp P complete problem so, the power of interaction is way more than NP or co-NP or sigma 2 or something the sharp 3 SAT is much much above that we saw the sharp 3 SAT as a public coin interactive proof system which used public coins to a very telling very nicely using public coins the this proof system could convince the verifier that the number of satisfying assignments to Boolean formula is equal to something which is a sum check protocol.

And I also mentioned the fact that IP is equal to P space. So, the P space complete language TQBF can also be shown to have an interactive true system. So, in fact that the same proof system that we use for a sharp 3 SAT can be modified to get a interactive proof system for TQPF. So, basically we have to add deal with how the existential and universal quantifiers are going to be converted into arithmetized into the arithmetic setting.

And some additional ingredients are required which I did not get into. I strongly recommend reading this article which say somewhat of an informal article and it just gives you a nice overview of the way the area had developed in the 80’s and maybe early 90’s. After that we saw public coin proof systems AM and MA. We saw that AM and MA are the only 2 public coins proof systems and we saw that AM k reduces to am which is AM of with 2, 2 rounds.

We saw that MA is contained in am we saw that graph isomorphism is unlikely to be NP complete because graph isomorphism if it is NP complete then sigma 2 equal to N pi 2 and the whole polynomial hierarchy collapses to sigma 2. One point that I missed to note when I stated this is that this is a nice theorem that graph isomorphism is unlikely to be NP complete but then if you look at this theorem closely you see that this the statement of this theorem just the statement graph isomorphism is NP complete.

Then sigma 2 is equal to pi 2 this has nothing to do with interactive proofs I could have stated this theorem at the time when we saw polynomial hierarchy and you would have understood this theorem we do not need to know interactive proofs to understand the statement of this theorem but this proof or this statement has a proof using interactive rules and and as far as I know whatever proofs are like there are small way the proofs that are there for this statement are all using interactive proofs may be slight variations of one another if you look at one reference from the other.

But then this the statement itself is completely a statement only on sigma 2 pi 2 and graph isomorphism it has nothing to do with interactive rules. So, it is very interesting to see that interactive proof systems the technology of interactive pro systems is able to show statements like this which have which are completely free of interactive groups. So, this is something that is very interesting.

And you could say that this is one reason why it was worth pursuing the interactive proofs and we saw that graph is unlikely to be NP complete. And then we saw that you can you can simulate public coin proof systems sorry private coin pro true systems with public coins and we saw. So, we did not see the entire Goldwarres and Sipser’s theorem but we saw the graph non-isomorphism as a public coin proof system.

So, what we had seen was a private coin proof system. So, this is the set lower bound protocol by Goldwasser and Sipser which used the fact that in the yes and since in no instance the size of the range of the range of messages that could be sent by the verifier could be vastly different and this is exploited in this proof system and one final point that I wanted to make is that there is something called probabilistically checkable proofs which is an area that that got created from the interactive proof systems that was an offshoot of interactive flow systems where you can.

So, it is like an interactive proof system where you can convince where the prover and verifier convince the prover convinces the verifier of some fact and the verifier does not have to look at the entire proof, the verifier just looks at some sub part of the proof. And this was also a success of the interactive proof systems. And this theory of PCPs or the probabilistic problems probabilistically checkable groups later was pursued in the 90s and early 2000s to prove a lot of results in the area of the hardness of approximation.

So, I will not get into what the hardness of approximation is because then it gets to too much. So, so the inability to approximate certain languages or functions and so, this is the this was an offshoot of the interactive roof systems and if you read the brief history of PCPs by Ryan O Donald that I mentioned in the previous part. One of the previous parts you will see this as well it is actually for this is the PCP that he refers to that.

Brief the history of the PCP theory and with that we conclude this week's content on interactive proofs and this being week 12, I would also like to summarize the entire what we have seen over the over the entire 12 weeks which I will do in the concluding video the concluding lecture, next lecture, thank you.