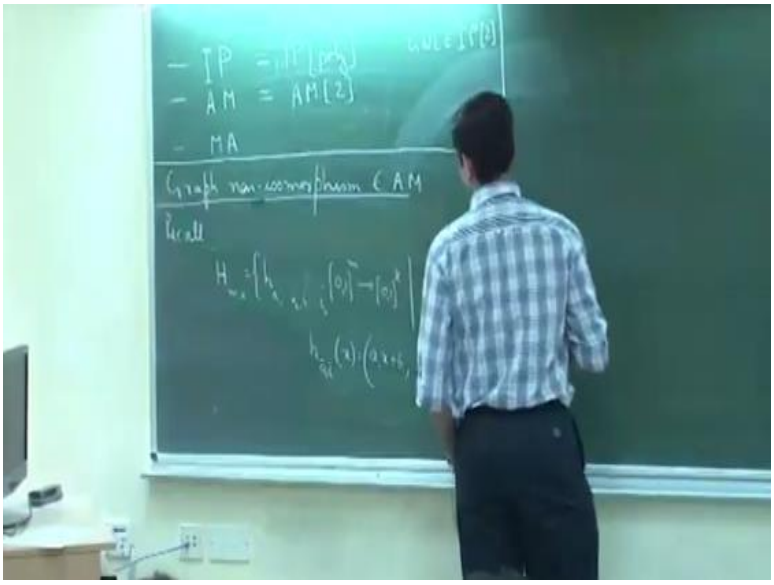
Computational Complexity Theory
Prof. Raghunath Tewari
Department of Computer Science and Engineering
Indian Institute of Technology, Kanpur

Lecture -32
Introduction

So, they both have some input and before the verifier asks a question or before the verifier sends its random bits, the prover gives a certificate, and then the verifier uses its probabilistic power and decides whether to accept the input or not.

(Refer Slide Time: 00:36)



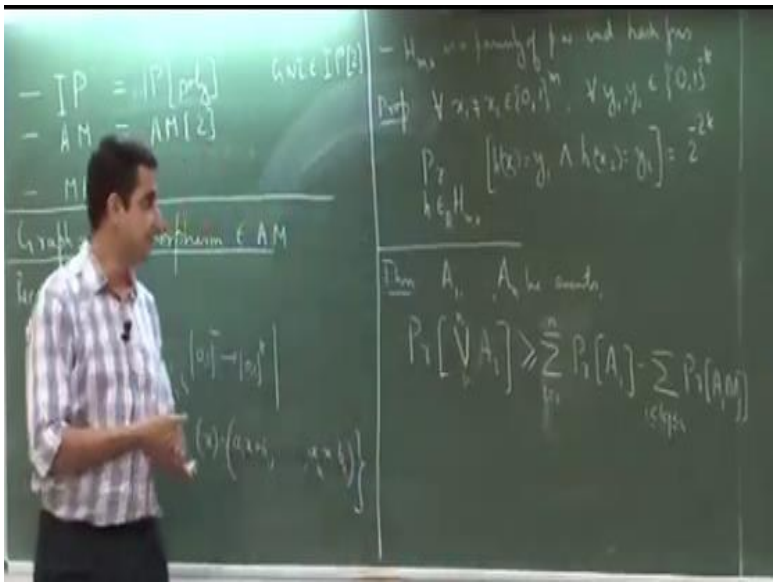
So, this is pretty much what we have defined so, far and we also saw last time that this graph non-isomorphism problem is in IP 2 in particular. So what we will see today the first thing is we give an AM protocol for the graph non-isomorphism problem. And the reason why this is very important this proof is because this actually gives the idea as to how to convert any IP kind of a proof to an AM kind of a proof, without simulate private random bits with public random bits.

So, that is the reason, so although we are doing it for one particular problem but the essential idea is very general. So let us see this proof this is interesting. So, first recall the following set of functions that we talked about a while back, when we were talking about brilliant Fazlani's

theorem so this class H mk, so class of pair wise independent hash function, so H mk is the set of all functions h small x, so then just index them with there are some case brings a 1 to a k;

And k bits b 1 to b k and h is a function from 0 1 to the power m 2, 0 1 to the power k, such that so h you can just denote this as a vector a, h k b of x is k1 dot x plus b 1 and so on till a k x + b k.

(Refer Slide Time: 03:59)



So, this is a earlier that H mk is a family of pair wise independent hash functions. So, actually a proof of this is also given in your text it is very easy I mean it is simple of algebra but you can look at this chapter eight to get a proof that why is this family pair wise independent. So, basically, when we say it is pair wise independent it means that it satisfies the property that for all x 1 not equal to x 2 in the domain.

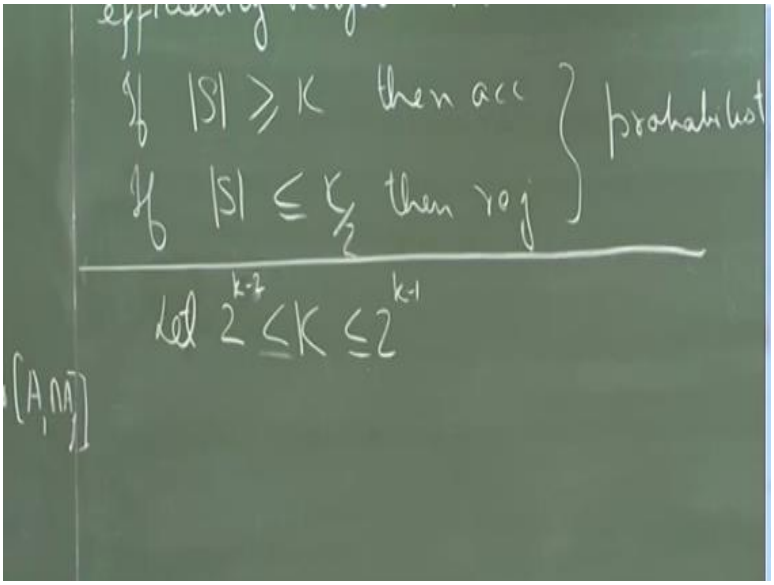
So, for two different strings x 1 and x 2 and for all y 1, y 2 so they need not be different, the probability that if you pick a function at random from this family that h of x 1 is equal to y 1 and h of x 2 is equal to y 2 is exactly 2 to the power minus 2 k, so basically its uniformly distributed in the in the range. So, we will use this fact and we will also use another tool from probability. How many of you here have seen the inclusion exclusion principle, you have seen it you guys.

So, but let me just recall the not the full inclusion exclusion principle, but the corollary of it that we will be using so if we have events A 1 through A n, so there are some n events so then what

the inclusion exclusion principle says is that the probability of all of them happening or, I mean the probability that some of I mean at least one of the event happens is; So, what we can write is write this as so this is greater than or equal to the sum of the probability of the individual events minus summation over for two distinct i, j's, the probability that both the events happen.

So, in its full generality, the inclusion exclusion principle basically just carries on this chain and this gets replaced with an equality but we will just use this corollary. So, that is what we will need so let us come back to the, so before we get into the problem we will actually solve another problem first.

(Refer Slide Time: 08:12)



So, this is known as the Set lower Bound Protocol, so let me state what the problem is so let S be some subset of 0, 1 to the power m, it consists of m bit strings and such that membership in s can be efficiently verified, so basically it means that so if there is a so when I say efficient here, I mean polynomial time there is a polynomial time verifier who given some let us say string of length m together with a certificate can very easily verify whether that string belongs to that set or not.

So you can think of this as a all m bit Boolean formulas, so together with an satisfying assignment it is very easy to check whether that formula is satisfiable or not. And let k be an integer, positive integer. So, the idea is as follows, so let me say what we are wanting here? So,

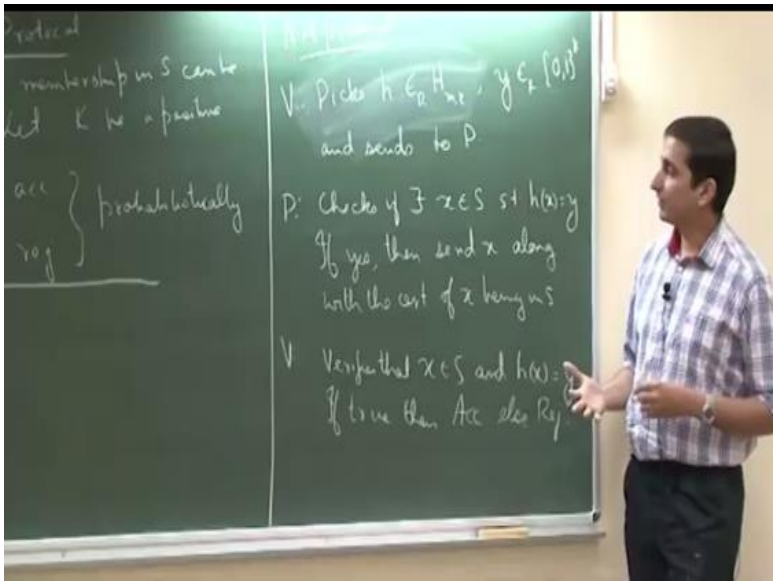
what we want is to approximate the size of S. So, in general S can be very large and as you can see.

I mean you need probably and not even an np kind more a machine which is more powerful than np to decide what is the size of S. Something like a sharp P kind of a function, but I do not want to exactly know what the size of S is, I want to approximately find out what the size of S is and more formally, what I want to find out is if size of S is greater than k then accept and if S is smaller than k by 2 then reject.

And we allow a probabilistic machine, probabilistically, so it can accept and reject with a certain probability. So, this is what we want we want to establish a gap between the size of I mean whatever the machine tells us, so if s is larger than a certain number it would accept and if it smaller then it would reject. So, how do we get a AM protocol for this problem, so the AM protocol is as follows.

So, before giving the protocol, let this number k lie between sum 2 to the power k - 1 and k - 2 and 2 to the power k- 1.

(Refer Slide Time: 12:34)



The AM protocol is as follows, so what the verifier does is, it picks an h randomly from this family, H m, k and it also picks a y randomly from the range or the codomains and sends to the

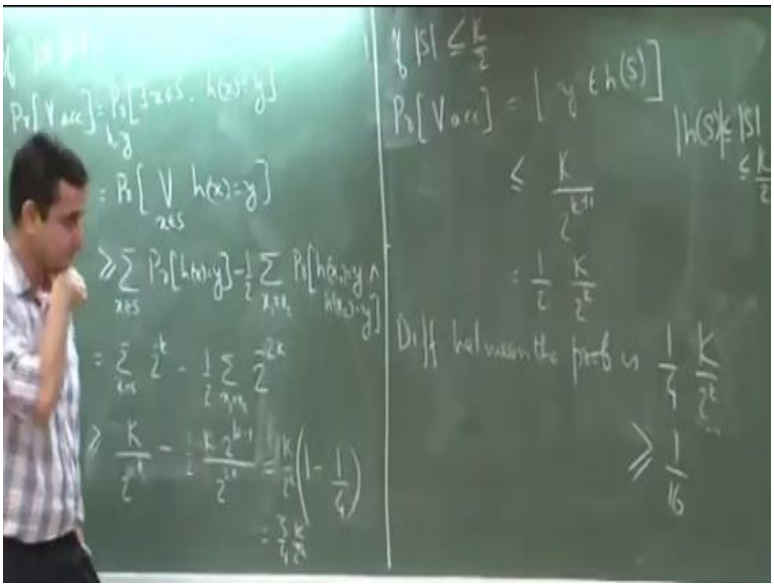
prover. What the prover does is? So, the prover wants to basically check that is there an x belonging to the set such that h of x is equal to y, is there an x which would hash to this element with respect to this hash function.

So, checks if there exist an x in s such that h of x is equal to y, if yes then send x along with the certificate of x being in S. So, note that S has this property that membership can be efficiently verified, so there exists some certificate which verifies membership that x is in s, so if there is such an x such that x belongs to S and it hashes to y so then the prover sends that x together with that certificate.

Now what the verifier does is? It verifies whether what the prover is saying is true or not. so verifies that x actually belongs to S and h of x is equal to y if true if both the conditions are true then accept else reject. So, let us see why this works? So, actually what we; will show is not exactly the definition of AM, so if you recall am says that it should for x belonging to l and not belonging to l.

The verifier should accept with probabilities greater than two third and less than one third respectively, but what we would show is a little bit something little bit weaker that the difference between these two probabilities is some constant, it is not exactly that the difference is one third but something smaller but that can easily be amplified. So, we will see that later on, but let us see the analysis.

(Refer Slide Time: 16:33)



So, again the analysis is not very difficult, so if S is greater than K, the probability that V accepts is, so what is the probability? It is same as writing that there exists something like that, so there exist and x in s such that h of x is equal to y, so this probability is over the choice of h and the random string y. So, this can be, so this is basically or of certain number of events so this can be written as the probability that h of x is equal to y for x belonging to S.

So, now we use the inclusion exclusion principle, so how can we bound this as minus, so a small error because we are over counting here because this says that i should be less than j, so that is for distinct i, so we have to put a half 2. So, this is what, so what is this probability? What is the probability that randomly if you have random functions h and random string y x will get mapped to y.

So, it is basically whatever is the size of the set that y is contained in its 2 to the power minus k minus half 2k. So, now it is just a matter of calculation, so this is the size of S so cardinality of S is how much is greater than k, so this I can write this as being greater than k by 2 to the power - k - half of into S - 1, so that will be 2 to the power 2 k divided by size of S into size of S minus 1 which is about k times I can write this as 2 to the power k - 1.

So, this is greater than k by 2 to the power k is how much can you will come to that later. I will just take k by 2 to the power k common so that is 1 minus 1 fourth, so you can verify this I mean

this is not difficult so that is three fourth k by 2 to the power k. So, let us just keep it at this point this one this is an equality, so the other direction is much easier so if S is less than k by 2 then what we know is so what is the probability that V accepts.

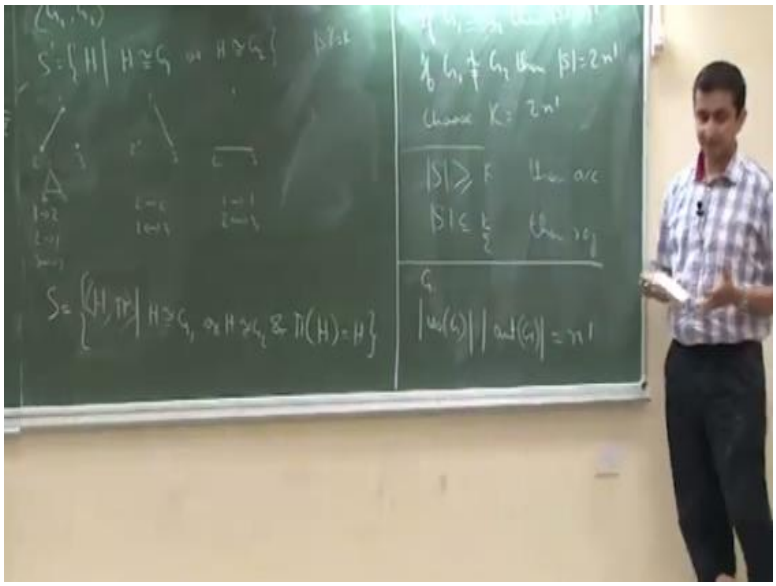
So, V accepts the probability is that y belongs to h of S, so this is less than or equal to so what is the set from which y can be chosen I mean the size of the set is 2 to the power k and the number of possible y's that I can have is the size of this set h of S but again h of S is less than or equal to the size of S which is less than or equal to k by 2. So, this is less than or equal to k by 2 just put a 1 here, so this is equal to half of k by 2 to the power k.

So, now we can see that the difference between these two probabilities is difference between the probabilities is one fourth k by 2 to the power between 3 4 and half. So, k by 2 to the power k is at least one fourth from this, so k by 2 to the power k is greater than one fourth from this inequality. So, therefore the probability is and the difference between the probabilities is greater than one over sixty which is sufficient for us.

So, how can you boost this probability I mean so that it falls on either side of one third and two third, so what the verifier can do here is that he does not pick just one but appropriately he picks some let us say some polynomial number of hash functions and corresponding to each hash function he also picks that many number of y's and sends them to the prover, so now the goal of the prover is to find one x which belongs to S and h i of x is equal to y i;

For all those possible i’s, if he can find such an x then he sends that x along with the certificate and now the verifier verifies so that will give a very large acceptance probability. But so is this clear to everybody, so now let us see how to decide the graph isomorphism problem or the graph non isomorphism problem using this protocol.

(Refer Slide Time: 25:30)



So, we will basically model the graph non isomorphism problem in a very clever way and that is the key behind solving this. So, we are given two graphs G 1 and G 2 and we need to check if they are non isomorphic. So consider the following set let me call this S prime, so let us say this is the set of all graphs H such that H is either isomorphic to G1 or H is isomorphic to G2. So, what we want is between these two cases?

So, if G1 and G2 are isomorphic and if G1 and G2 are not isomorphic between these two cases the size of this S should have a large difference, so what can we say here so ideally what we want is that, it is not exclusive, it can be that and for example if G 1 is isomorphic to G 2 then its basically all graphs which are isomorphic to either of them. Suppose if it is isomorphic to G 1 let us say no that is not true actually here.

So, that is what we want, that is what we want that this the difference between these two sizes should be a multiplicative factor of 2, but here that does not happen and the simple reason for that is consider the following example, so we have let us say the following graph a three vertex graph as follows, so what are the graphs that are isomorphic to this so we have this graph we have a this graph and we have this graph.

So, although there are three factorial possibilities, but there are only in I mean in this case which should be 6 but there are only 3 graphs that are distinct and isomorphic to the given graph. But

together with the isomorphic graph if you also consider what is known as an automorphism, so an automorphism is basically a permutation which maps to the same set. It is basically a permutation.

So, suppose, here if I consider the permutation which maps 1 to 2, 2 to 1 and 3 to 3. Here if I consider the permutation 2 goes to 2 and 1 and 3 gets interchanged, so here 2 and 3 gets interchanged and 1 goes to 1. So, together with H, if I also consider all permutations pi such that H is either isomorphic to G1 or h is isomorphic to G2 and pi is a is an automorphism on h so pi of H exactly gives the graph h.

For example here if you apply this automorphism, it will exactly give you the same graph, so if I consider this as my set then what you are claiming is true so then what we would have is that the size of s is so then if G1 is isomorphic to G2 then size of S is exactly n factorial and if G1 is not isomorphic to G2 then the size of S is 2 n factorial, but all I am saying is that the point why I is this example is if I just take this as my set then this property need not be true.

There need not be a gap of n factorial but, so but we can always pick this so now what we will do is we will just consider this set in our set lower bound protocol, and it is easy to see that membership in S can be efficiently verified, because if you have a graph h and pi what would be a certificate that would be able to verify membership efficiently. So, G 1 and G 2 are two fixed graphs, so that is known to the turing machine the np machine.

So, now suppose it gets a pair H, pi how does it decide whether it belongs to the set s or not, so be together with H, pi the certificate that is given is either a permutation from H to G 1 or a map from H to G 2, and it is also told that what kind of a map it is or maybe that is not necessary also. So, together with that map this is very easy to check. So, now the k that we consider is, so, this is the; set S and choose K equal to what?

So, what is the k that we would want to fix? 3 by 2 into n factorial that will not help no, because k greater than 3 by 2 into n factorial does not still imply that G 1 and G 2 are non isomorphic. So, we will just pick this as 2 n factorial, because greater than equal to means that it at least should

be equal and that is it, so we will fix this as our set that is there and we will pick k to be 2 n factorial.

No, we cannot do it that because the difference is not known to have a multiplicative difference of factor 2 here it is only when we club this together with automorphisms, then this very nice explicit characterization is known so this is a very nice thing I mean you model the graph isomorphism problem into a totally different problem and then you show that you do have a public coin protocol for it.

Because see what we want is so what was the protocol for? So, what the protocol says is that if S has size greater than or equal to k then it accepts and if S has size less than k by 2 then reject that was the set lower bound protocol, that is what we are doing. So, now suppose this is our k so if that size S has cardinality greater than k then it means that G 1 and G 2 are non-isomorphic, so then I am accepting with high probability.

On the other hand if S, S i smaller than k by 2 which is less than or equal to n factorial then I know that they are isomorphic with again high probability. Then half, no then it is not known I do not know but it should be I mean that is a nice thing to think about, so suppose the gap is less than I mean let us say some c times k where c is a constant that is less than half maybe I do not know I do not might help.

I mean there is no reason why it should not help I guess but yeah you can think about them, so basically what we argued earlier was that suppose if there is some set which is implicitly known, so there is some set that is implicitly known for which membership can be efficiently certified. So, then approximating the size of that set can be done by an am protocol that is what we claimed earlier and the way we can do it is using that protocol.

So, if and when we say approximating that said we mean this precise formulation, so if s has size greater than k then we accept and if it has size less than k by two then we reject, so we are not even getting the size back we are getting some kind of a very crude approximation. So, this is

what that earlier AM protocol allowed us to do. Now, what we claim is that the graph non isomorphism problem can be modeled as a instance of that set lower bound problem.

And the way to model it is that to construct a set which has this property that if G 1 and G 2 are isomorphic then the size of that set is some value let us say m and if they are non isomorphic then it is some value 2m. So, how do we construct such a set so that is the goal so one obvious thing to try for is something like this, if I consider the union or if I consider the class of a set of all graphs which are defined this way.

But the point is that, this will not give a characterization of that form, So, the gap will not be m and I mean the two sets will not have sizes m and 2m but what will allow us to have is sets of size m and 2m is if we model the set in this fashion, and then once we have sets which have size m and 2m in this case n factorial and 2n factorial we just set that k to be our two n factorial and run the whole protocol.

Which transformation, here we are not doing a transformation I am just saying that this is not a correct set for us and the reason why this is not a correct set is, so I want to construct a counter example basically so if I pick this as my definition then the different the true sets will not have the appropriate sizes. So, suppose G 1 and G 2 are indeed isomorphic and let us say that this is my G1.

So, the set of h that is isomorphic to either G 1 and G 2 is basically just these three graphs, so this set has size 3 now suppose G 1 and G 2 are not isomorphic, So, let us say that G 1 is this graph and G 2 is this graph or no not this graph let us say that G 2 is this graph. So, then what is the set of graphs that are isomorphic to either G 1 and G 2 you have these three graphs and the only graph that is isomorphic to this is this so you have only four graphs.

So, which is three plus one that is why you will get a problem, but now once you allow this automorphism together with this here what you have is that although there is only one graph which is isomorphic to this but there are three automorphisms of this, so if I have a vertex 1,2

and 3 here. So, 1 can be mapped to 2, 2 can be mapped to 3, 3 can be mapped to 1, I can also have one can be mapped to three 3 to 2, 2 to 1 and there will be a total of 6 of them n factorial.

So, you can prove it I mean the proof is also not difficult so what you can prove is the following that if you look at any graph G and if you look at the set of all isomorphic graphs of G. So, let set of all graphs that are isomorphic to G so this times the set of all graphs that are automorphisms of g is exactly equal to n factorial. So, again it took as a simple example. So, if we consider this graph the only graph that is isomorphic to this is itself.

So the first set has size 1 but the second set will have size 6, here there are 3 graphs which are isomorphic to G but the number of automorphisms of this is exactly two because the only way to get an automorphism is three gaps mapped to three and one and two gets interchanged so that is three times two. So, I do not want to get into the details of this so this is a kind of a basic fact from group theory but I will not go into the details of this.