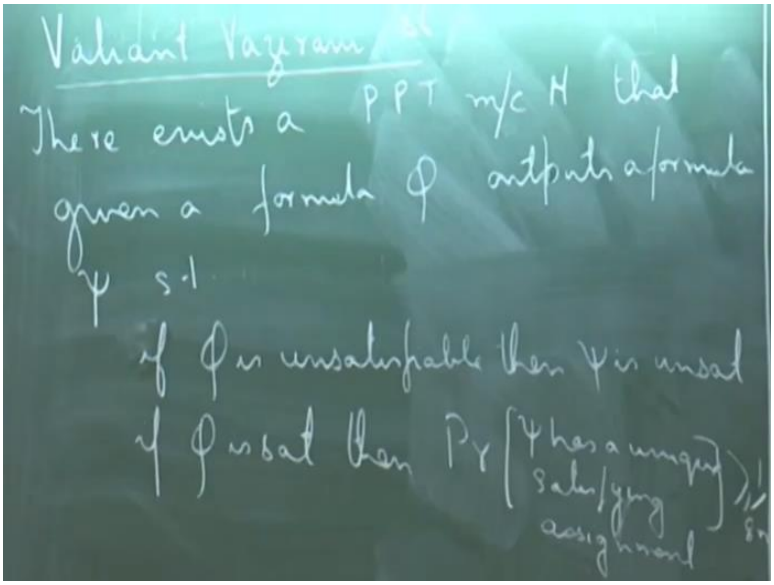
Computational Complexity Theory
Prof. Raghunath Tewari
Department of Computer Science and Engineering
Indian Institute of Technology, Kanpur

Module No # 05
Lecture No # 23
Valiant-Vazhirani Theorem - I

So the result that we are going to discuss today and we will look at some applications of this in our next class.

(Refer Slide Time: 00:27)



So this was due to 2 scientists Valiant and Vazirani can in fact this is the same Vazirani who came to our department I think last month and this was in 86 or 87 I think 86. So let me go ahead and state the result and then we will look at the necessary definitions and how the proof goes. So what this result says is, there exists a probabilistic polynomial time machine M that given a formula.

So it is a Boolean formula Phi outputs a formula let us call it Psi such that if Phi is unsatisfiable then Psi is also unsatisfiable and if Phi is satisfiable then with high probability. So actually probability greater than 1 over 8 n Psi has a unique satisfying assignment. So basically given a Boolean formula in lets in CNF form we will construct another Boolean formula which with very high probability will have a unique satisfying assignment.

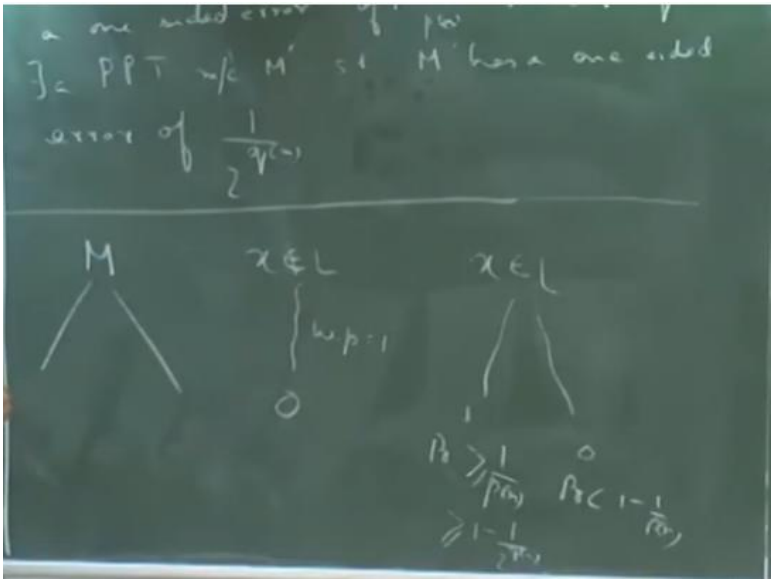
If my original formula was satisfiable and otherwise it does not have any satisfying assignment. So this is the main technical Lemma that we will prove today and may be Monday we will see some nice applications of this theorem because we can anyway amplifying this is not difficult. So note that this is so that was actually going to be my next topic but note that this is an algorithm which has a one sided error.

So it is like an RP algorithm not technically RP algorithm because it is not giving a Boolean output. It is outputting a string actually but it is having a one sided error so it is very easy to amplify the success probability of this algorithm. In other words how do amplify success probability so we saw how to amplify the success probability of a BPP algorithm and that was kind of technical because we needed all these fancy bounds to get the result done.

But if you have one sided error then it is much simpler to amplify the probability. Because what you do is you take let us say some t trails of this algorithm and even if in one of the trails you get a accepting I mean if the machine accepts or if the machine correctly outputs a formula in this case then you go ahead and output that formula. So what is the probability that so the probability that it gives a wrong answer is 1 - 1 over 8 n whole raise to the part t.

And that can be brought to less than half or less than any polynomial for t. Which is at most polynomially large so that is not difficult to see? So let m write that in because I was planning to do.

(Refer Slide Time: 05:45)



So what we can show is just to rephrase what I said in terms of an RP machine. Let M be a probabilistic polynomial time machine with a one sided error of let us say some over some polynomial p n then for all polynomials q there exists a probabilistic polynomial time machine M prime such that M prime has a one sided error of 1- error. So here I was talking about error so I will put this as 1 - 1 over pn and this has error of 1 over 2 to the power q of n.

So even if you; have a very large error but still polynomially bounded away from one some inverse polynomially bounded away from 1. It can be reduced to an exponentially low error probability and this is just by taking some polynomial number of runs of this machine M and if the machine accepts in any of those runs you accept. This is not very difficult to show. So that is the reason why we consider this probability as sufficiently high success probability.

So any questions about the statement or you mean to say why is this? No I am not saying this is easier to handle but point is that this has some nice properties. So this formula has a property with that with high probability it has a unique satisfying assignment so that helps us in certain cases. As I said we will see applications of this result may be next class so we will see why this is a nice thing to have?

But because this is not very this is something not obvious I mean you have a status I mean you have a Boolean formula which can have any number of satisfying assignments but you are claiming that without making any large error. So without making only some small amount of

error you are reducing it to another Boolean formula which has only one satisfying assignment. So that is quite a nice thing to happen.

What you mean binary this machine? So you can look at the length of this whatever is thing string you can look at the length of this string suppose this has some length n and let us say this comes from a universe which has length m. So you get all functions from n to m such that if this is a formula which is satisfiable then the resultant string will give you a formula which is which has a unique satisfying assignment with high probability the resulting string.

Will corresponded to the formula and if this is not satisfyable then it is not here. Yes n digit of what? So you can formulate that as a so no but that i think you are talking about languages. So languages you can think of them is Boolean functions if you have a language you can think of that as a Boolean function which maps to 1 if the string is present in the language and the maps to 0 otherwise.

So what do you mean by reduction from a string to another string? You can only reduce to one language to another yes correct 1 bit same way you can do this also. So this machine you can think of this machine as a collection of all those m functions each of which is giving 1 bit of the string Psi but this is not just any obituary function. This is a randomized function in same sense. So for the some string it can give a 0 bit in some cases and it can give a 1 bit.

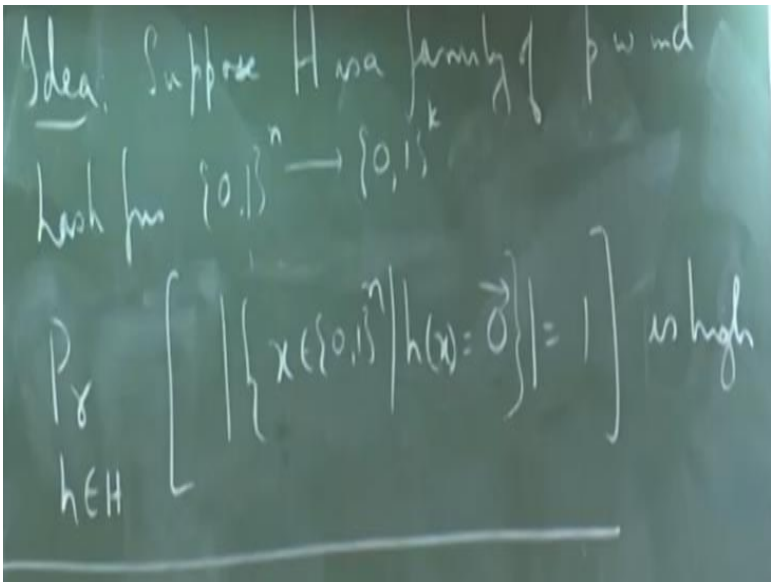
In some cases it is not deterministic function because the machine itself is randomized. So if the concatenation of all those values satisfies this property that the string that you get has this property that it is it has unique satisfying assignment then in think it will get incorporated. Individually you will not get any data I mean individually is just one bit and you want to look at the concatenation of all these so any other; questions?

So here what I am saying is that so suppose you have a machine m which has one sided error. So if it is given in no instance suppose you look at a string x which does not belong to the language then it will output 0 with probability equal to 1 and if you give it a string x which is in the language then it outputs 0 with probability less than 1 - 1 over some p of n. And it will output 1 with probability greater than because this is not a good way to write this.

So it outputs 1 with probability greater than 1 over p of n and 0 with probability less than 1 - 1 over p of n. So that is what I am saying suppose if we have a machine which has this one sided error property. So here note that this probability is quite high and it is giving a wrong answer with may be a large probability but it has the nice probability that it is only a one sided error. So here I am saying that from this machine you can get another machine m prime which has the same thing.

So if the string is not in the language it gives 0 but if the string is in the language. So this probability gets amplified to greater than 1 - 1 over 2 to the power q of n. So from 1 over p of n think of p of n is something like 1 tenth. Earlier it was giving only let us say probability one tenth of success. Now it gets amplified to something very large for any polynomial q of n. And his can be achieved very easily by taking m runs and taking yes in any one of those runs giving yes answer.

(Refer Slide Time: 15:27)

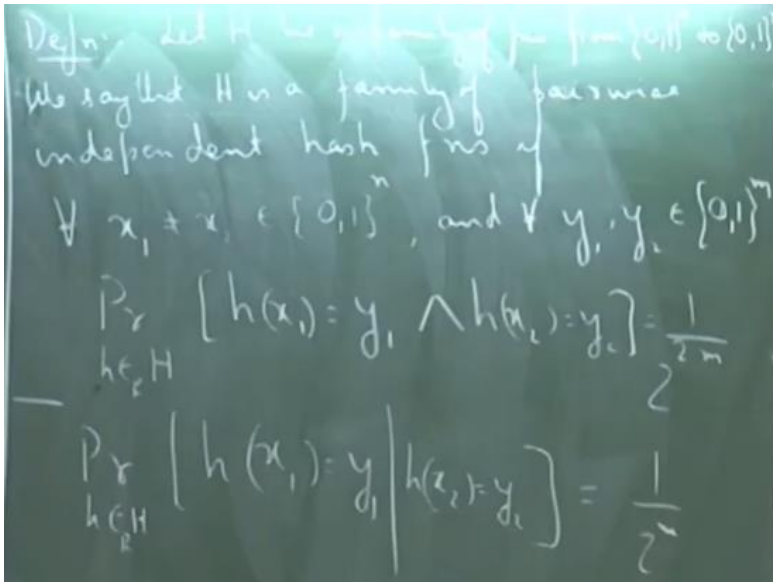


So let us see what is the idea behind this theorem first? And then we will prove it. So the idea is that so suppose h is a family of pair wise independent hash functions. Let us say from 0 1 to power n to let us say 0 1 to the power k. So what we mean by this pair wise independent has hash functions will come in a minute but basically it means that all factions in this family distributes their output strings quite nicely in the set 0 1 to the power k.

Then a probability that if I pick a random h here the size of this set is one so this set is the collection of all, those x as such that h sends those x to the all 0's vector. So this probability is high. So if I have this nice family of functions then the probability that a randomly chosen function. I mean for a randomly chosen function the number of vectors which get mapped to the all 0's string.

So that is equal to 1 has a high probability. So this is what the main idea behind this theorem and this is what we will formulize and prove. No I mean here n and k can be any general numbers. So what this says is that for such a function on an average there is only one string which will get mapped to the all 0 string. So let us look at the proof before that let us look at some questions.

(Refer Slide Time: 18:14)



The first definition is regarding this hash functions. So let h be a family of functions from n bit strings to m bit strings. So then we say that h is a family of pair wise independent hash functions if for all x 1 not equal to x 2 in your domain and for all y 1, y 2 in your co domain. The probability that a randomly chosen function h has the property that h of x 1 = y 1 and h of x 2= y 2 = 1 over 2 to the power 2 m.

So what this I mean if x 1 and x 2 are the same then it implies the y 1 and y 2 are the same. So it does not make any sense. So if I have 2 points so what this says is that suppose if I look at this picture. So suppose this is my domain and this is my ring so if I pick any x 1 here and I pick a

random function here so what is the I mean what do we mean when we say that this function randomly maps this x to 1 point in the co domain.

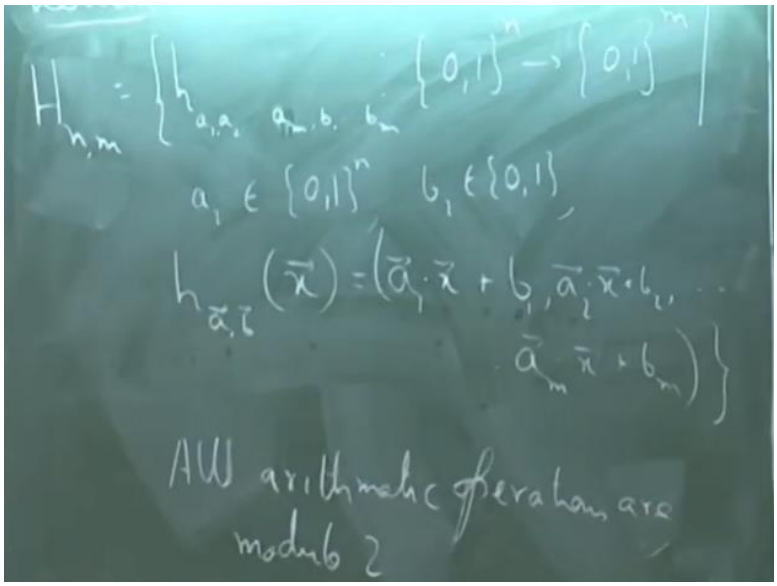
It means that for any chosen y here h will maps x to, y with probability one over whatever is the size of the set that is what it would mean. But here I want something more I mean I am not only satisfied with getting this kind of independence. What I want is for any 2 strings here so if I have any 2 string x 1 and x 2 which are different and I pick any 2 points in my co domain. The event, that x 1 gets mapped to y 1. So this is my y 1 and x 2 gets mapped to y 2 these 2 events are completely independent.

So basically the probability that one of them happens is one over 2 to the power m so because the size of this set is to the power m and since they are independent the probability is 1 over 2 to the power 2 n. So that is what it means so if this happens then i call such a family as a pair wise independent hash function family. So one other way to interpret this as so just an alternative interpretation this means that the probability that if I pick an h randomly from H the probability that h of x 1 = y 1.

Given that h of x 2= y 2 should be what do you expect it should be? It should just 1 over 2 to the power m I mean I other word it should not depend on what this h is doing for this point x 2. Yes not necessarily so if i allow y 1 and y 2 to be different then i will rise I mean I might have a problem because the first y 1 can be anything but for the next value of phi which is y 2. You are only allowing it a set of size 2 to the power of m - 1. So of course that probability decrease.

So it is not necessary that y 1 and y 2 are different they can be the same. So now the question is that does; such families of functions exist. I mean so this is just a hypothetical family is there an explicit way to construct such a family.

(Refer Slide Time: 23:55)



So the answer is yes and I can state this as a Lemma and I leave the proof as exercise. So let us define this family h before I go to that no but see what do you mean by all possible hash functions? You cannot I mean what do you mean by a function being a hash function that is not making sense. I mean you can only talk about a family being a collection of hash functions. If you look at the set of all possible functions then clearly it does not satisfy this property.

Because if you look at the set of all possible functions from n bits to m bits. I am not sure may be it will may be it will be a family of hash functions. But see ideally we want 2 things ideally we want a collection of functions to be. I mean to have this property and also we want this family to not have a very large size. Because if I look at the collection of all possible functions from n to m.

So that is huge that is exponential in, n in fact x y is super exponential. So what I ideally want is a small collection of functions which will have this property. So probably you are right if you take the collection of all functions you can prove that it has this property. I am not sure but it should be true. But so the goal to define a small set of such functions so; that is what I am going to show here?

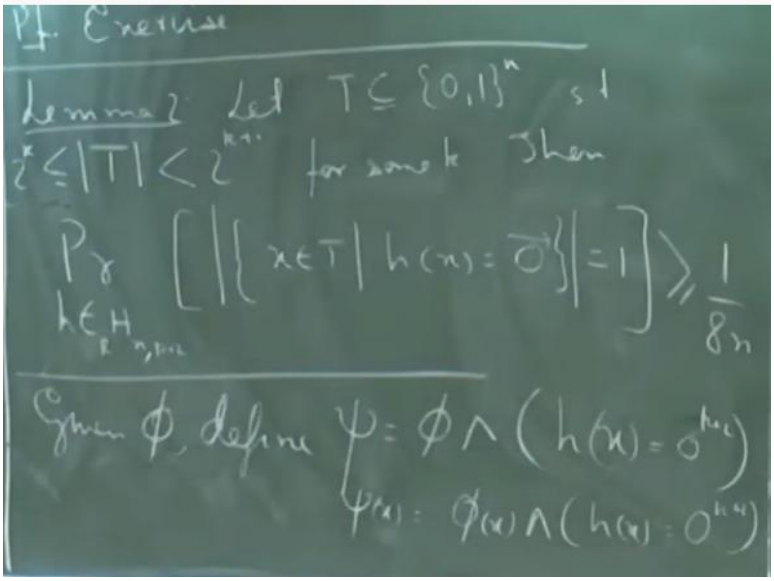
So what notation do I have so I will just use this notation h n, m this is the collection of all functions of the form h a 1 a 2 up to some a m b 1 to b m. So these ate functions from 0 1 to the

power m n 2 0 to the power m such that these a i. So these are n bit vectors the b i are just singe bit and h of say some a, b. So I am just denoting this as a vector of vectors given some vector x.

So this is equal to a dot x plus this is denote as a 1 x + b 1, 8 square + b 2 and so on up to a m x + b m. And all arithmetic operations are modular 2. So what i am doing is so all these functions output a m bit strings as follows. So I take the do product of a 1 with x and add the bit b 1. And similarly I do this for all the bits of my string h of x and that is how i get these m bits claim is that if you have these vectors a 1 through a m.

Which are n bit vectors and if you have some m bits b 1 through b m then the collection of functions that is defined by this definition. So this has the property that it is a pair wise independent hash function family.

(Refer Slide Time: 29:30)



So h n, m is a pair wise independent family of hash functions. So i will just leave this proof to you. This is a proof is little technical in nature because you have to show that it satisfies this property but it is not difficult. You can prove this. So please try this proof so is the definition of this set clear you look at all possible. So let us say you look at all possible n bit strings for your, a i and you look at all possible strings for this b.

So for one setting of these a i's and b i's you get one function. So this family will have 2 to the power m size which is not which is smaller than super exponential. So let us look at another

Lemma. So let p be a subset of 0 1 to the power n such that not equal to 2 to the power k greater than or equal to 2 to the power k and less than 2 to the power k + 1 or some k. Then the probability that if I; pick h randomly from this family of functions h n, k + 2.

This set of all x is belonging to t such that h of x is equal to the all 0's vector as cardinality one with probability greater than 1 over 8 n. So if we can establish this Lemma then note that because I rise the theorem. But that theorem will follow from this because then what you do is suppose we have suppose we know that this lemma is true you can randomly pick these strings a 1 through a m and b 1 through b m.

So that will give you a function here and then what you do so I just mentioned here then you look the conjunction of these 2 formulas. So given Phi you define this function Psi as Phi and whatever formula will represent this following fact that h of x is equal to the all 0's vector or more particularly h of x = 0 to power k + 2 k. So this is how you define Psi. So now suppose if this Lemma is true let us look at the 2 cases.

Suppose if Phi were to be unsatisfiabe then clearly Psi is also unsatisfiable because this is conjunction of some unsatisfiable formula and something else. So this can never be satisfiable but suppose if Phi were to be satisfiable then what we get from this Lemma is that, if I look at. If let t denote the set of all satisfying assignments of Phi then we know that there is a high probability that a randomly chosen hash function will map all, the x.

I am not I am sorry it will have the property that there is only one x which gets mapped to the all 0's vector. So therefore if I look at Psi of x this is equal to Phi of x and h of x = 0 to the power k by k + 2. So this will have just one satisfying assignment with probability greater than one over 8 of n. Because it is just that one string which will make this formula evaluate to true and of course that will also make this also evaluate to true. So; there will be multiple strings of Phi x.

There are multiple strings here but only one of those strings among them will be able to make this equal to true with high probability because that is what I pick. So here I am saying that so suppose this Lemma is true. So then this says that suppose I pick it set of strings such that it has the property that it lies between 2 to the power k n 2 to the power k + 1 then this happens. So what i do is that so.

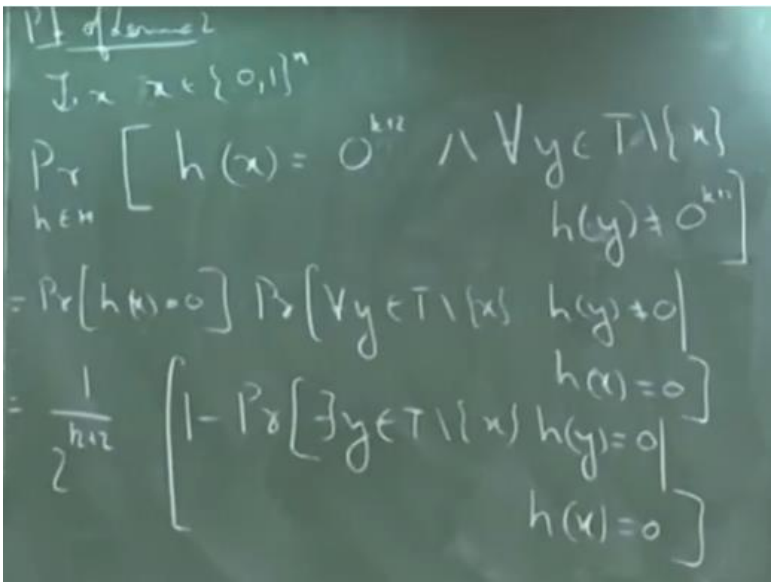
So we have actually I skipped a step here so given 5 what you do is that you guess this function h by guessing these a 1 through a m and b 1 through b m and you also guess a number k such that the number of satisfying assignments of phi lies between 2 to the power k and 2 to the power k + 1. So that also actually that was going to come here but anyway so you also guess that number k.

And then together with that you have the guarantee that t is the set of all satisfying assignments of phi and only one string within t will make this evaluate to true with high probability. No k is the k is this bound so 2 to the power k and 2 to the power k + 1 are bounds on the number of satisfying assignments of Phi. So there is some k such that that gets bounded right yes what do you mean?

So to think about it let me just present the proof of this Lemma you can think for it for a while we will come back to this later on. You want k to have this property actually when we look at the proof of this Lemma will need both these bounds. So 2 to the power k should be smaller than t and 2 to the power k + 1 should be larger than t. So that is so we just guess one of them so what we show is that suppose we have a fixed k then the probability that this happens is greater than one over 1 by 8.

But now if i am also guessing this number k the total number of possible choices for k is n. So this probability gets reduced by a factor of another one over n that is still fine with us. So what we will show initially is that suppose we are fixing a k so suppose we know what our value k is then we can argue that this probability is greater than 1 over 8. So let us look at the proof of Lemma 2 first.

(Refer Slide Time: 37:58)



No we; will not use Lemma 1 anymore so we are just using the fact that this is a family of hash functions. So Lemma 1 only gives you an explicit way to construct such a function. So when you write the actual algorithm for the theorem so when you have to design the probabilistic polynomial time algorithm how do you get the function? So you guess all these a i and b i and that gives you the hash function.

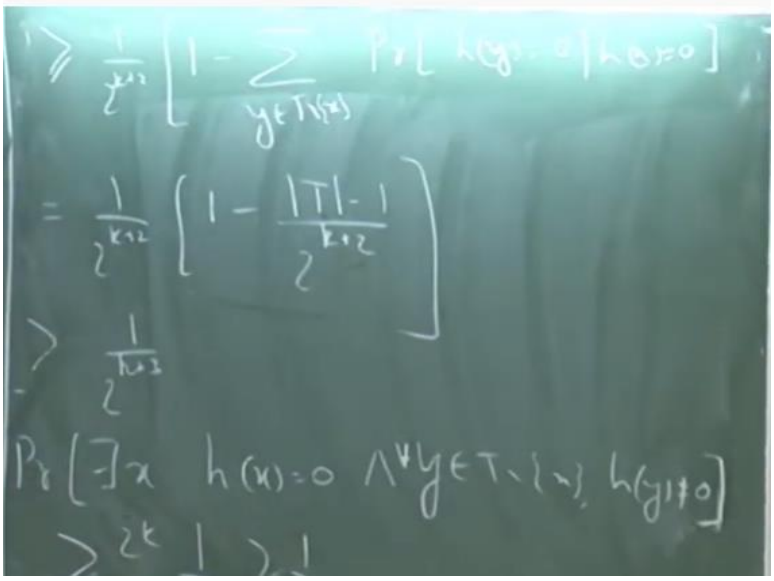
So that is where Lemma 1 is used so first let us fix some x 0 1 to the power n and let us also suppose that we know what our k is. Suppose that so we know what this particular case. So what we want to estimate is the probability or a randomly chosen h that this set has size 1. So how can we alternatively write this event so this can be written as h of x is equal to the all 0's vector and for all y in t - x h of y is not equal to the all 0's ,vector.

So this is another way of writing there is only one x which gets mapped to the all 0's thing so let us try to look at how this can be written so this can be written as the probability that h of x = 0 times using base theorem the probability that all y in t - x h of y is not equal to the 0's given that h of x = 0's. So i am just for case of notation just skipping this superscript of k + 2 in all the vectors.

So this is by base theorem so this probability is easy because this is a family of hash functions. So a fixed x gets mapped to some fixed y is what 1 by 2 to the power k + 2. So now the thing is that how do I estimate this probability? So what is the compliment of this event so the

compliment of this event is there exists a y in t – x. Such that h of y gets mapped to 0 given h of x also gets mapped to 0. so now we can use the union bound to bring that there exists outside.

(Refer Slide Time: 41:58)



So that is greater than or equal to 1 over 2 to the k + 2 1- sigma over all y in t minus of x the probability that h of y = 0 given h of x = 0. So I am just using the union bound here now what is this probability? So since h is a family of pair wise independent hash functions this is equal to 1 over 2 to the k +2 1- I am just skipping a step here. This set has size t - 1 and this probability is 2 to the power k + 2.

So this is greater than so now note that this probability is greater than half right because t is less than 2 to the power k + 1. So t - 1 2 to the power k + 2 is less than half. So 1- a quantity which is less than half is; of course great greater than half. So this is less than 2 to the power k + 3. So that is all now the question is that I had done this entire thing for fixed x. But basically what i need is that this will be strictly greater.

So the thing is that how many x do we have in t. So basically what we want is finally to get this probability and this is not quite the same as this because here we are fixing an x. So what we want is the probability that directly that there exist an x such that this entire thing happens h of x is 0 and for all y in -x h of y is not equal to 0. So how many x do we have? So we have a lower bound on the number of x which is 2 to the power.

So therefore this probability is greater than 2 to the power k times 1 over 2 to the power k + 3. So this is greater than 1 over 8 actually this is strictly greater than 1 over 8. Suppose we know what our k is then we have this to be strictly greater than 1 over 8. But now as I said that since we do not know what k is I can just randomly guess a k at the beginning. So that is why I get the extra 1 over n.

So let us stop here today we will look at the connection between this Lemma and how we get the actual theorem may be a little bit more beginning of next class. And then we will see applications.