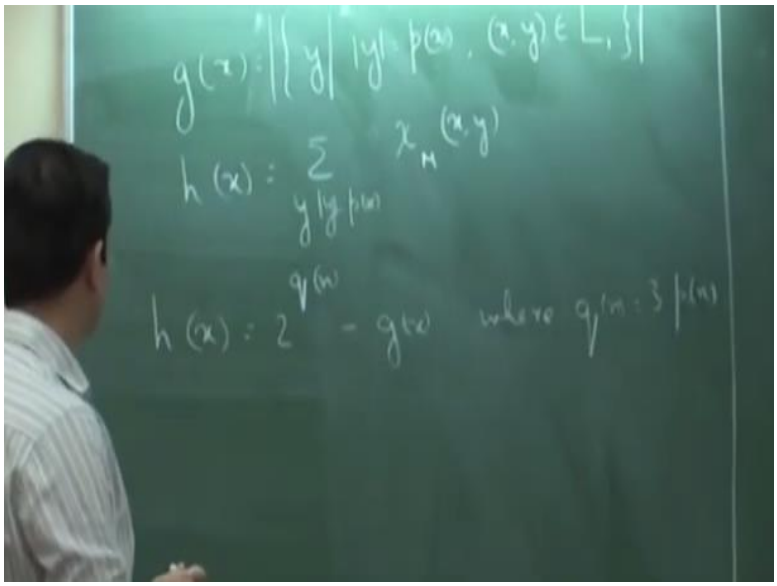
Computational Complexity Theory
Prof. Raghunath Tewari
Department of Computer Science and Engineering
Indian Institute of Technology, Kanpur

Lecture -28
Introduction

So, I will just revisit the conclusion of Toda's theorem. So, in last class we kind of quickly looked over y BP dot parity dot p is in p to the sharp p but let us just go over that argument once again.

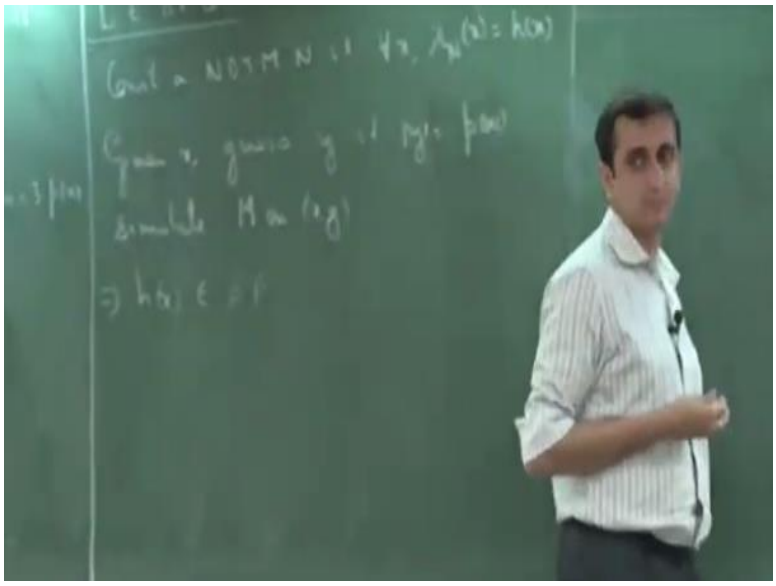
(Refer Slide Time: 00:42)



So, what we had where these two functions, we defined the function g of x as cardinality of the set of all certificates y. Such that y had length p of x and x, y belong to the language L 1. So, L 1 was that parity P language. And, we also had this function h of x which was the sum over all y's ok such that y has length p of x of the values chi m, x y. So, where this m was this parity P machine with its modulus boosted up to some 2 to the power q n.

And because of the way we define these two functions what we had was that h of x is equal to 2 to the power q n - g of x where this q basically comes from the polynomial p. So, I think we can, defined it as 3 p n, q of n is, so we had this so now the thing is how do, we get our p to the sharp p machine.

(Refer Slide Time: 03:10)



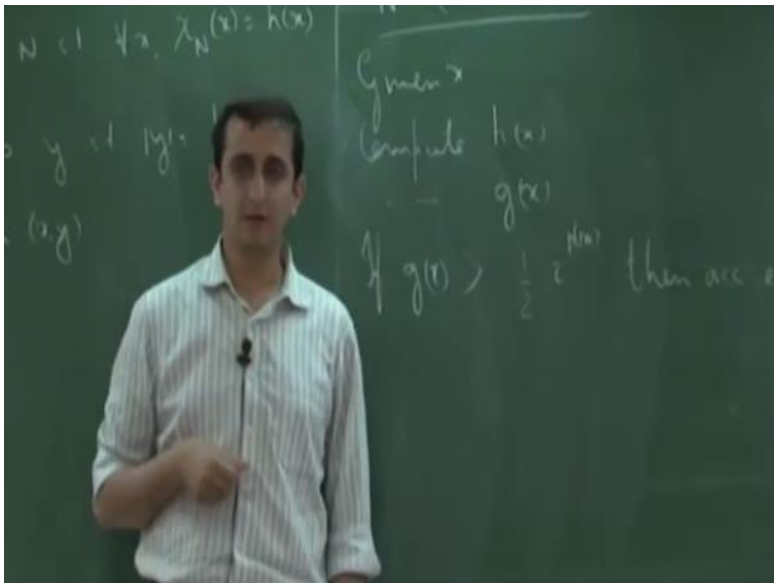
So, let L be a language in BP dot parity dot P. So, we want to show that this is in p to the sharp p, so given x what do we do, so we have this L. But I will not give the p to the sharp p machine right now. So, let us, so what was the notation n, so we construct a non deterministic machine N such that for all x chi N of x would be equal to our h of x, so this is what we want so how do we construct this machine?

So, given x, so h of x is just sum of all these terms. So, we guess a y and then simulate the machine m on x, y. So, now by definition this machine n will have h of x many accepting paths and since we have a non deterministic machine N which has h of x accepting paths so this implies that h of x is in sharp p. So, now this is basically the oracle that we will use to get our p to the sharp p, upper bound on L.

Yes, now that is what we set it to be so to boost the modulus so if you look at that lemma so last time when we gave that lemma, so that was for all polynomials q. So, for any polynomial q given any parity P language we can boost the modulus of that parity P language to 2 to the power q. So, we start with basically a BP machine, so the BP machine uses some number of random strings which is some polynomial.

So, let us say that polynomial is p, so given such a, p we choose our q of n to be 3 times p of n. So, that is what we discussed last time while giving the proof of this equality. So, y 3 to the power, I mean 3 times p n will suffice. So, what do we do now?

(Refer Slide Time: 06:48)



So, let us say we will construct a machine N tilde which is a, p to the power sharp p machine for L so given x, compute h of x by making a query to the oracle and from h of x compute g of x which is just a subtraction and now if once you know g of x, so g of x is the number of certificates of for a string x to belong to L 1 and since L 1 is a BP language the number of certificates is either greater than two thirds the number of total possible certificates or less than one third the number of total possible certificates.

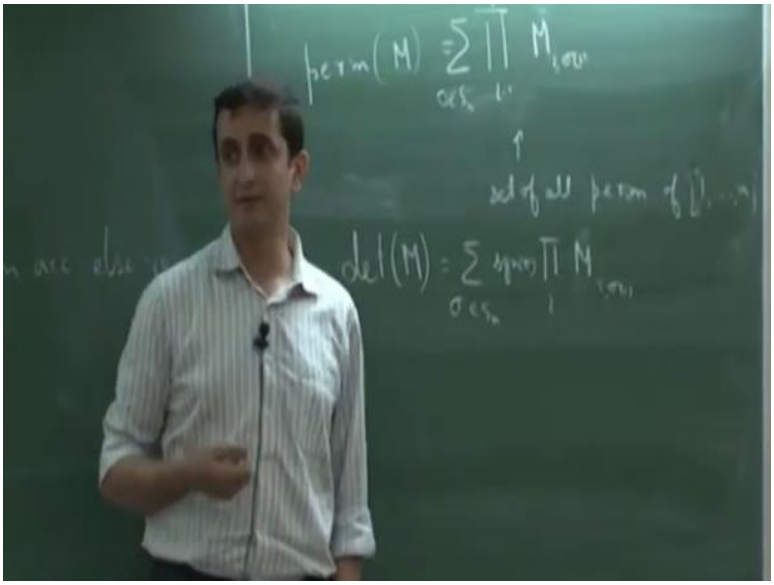
So, if, g of x is greater than let us say half times 2 to the power px then accept else reject. Because we have the promise, so that is all. So, any questions yeah, so capital M was that machine so the parity P machine with this boosted modulus. So, we saw that given any language in parity P we can and any for any polynomial q we can construct a machine m.

Which will have this property that if x belongs to L, then the number of accepting paths is congruent to minus 1 mod 2 to the power q and if x does not belong to L then the number of paths is congruent to 0 mod 2 to the power q. So, M is basically that machine. So, yeah, so I will,

so this completes our discussion on Toda's theorem. So, what we will do next is, that is a next week.

So, I already mentioned that the language sharp SAT is complete for sharp p and that basically follows from the observation that Cook's reduction is parsimonious. So, whenever given any input the Boolean formula that is constructed has the same number of satisfying assignments as the number of accepting parts of that input on the given non deterministic machine. So, we will also look at another very important sharp p problem which is known as the permanent problem.

(Refer Slide Time: 11:00)



So, I will just briefly define what that problem is, so let M be a n cross n matrix, so the permanent of M is defined as the product over all permutations of n, so sigma belonging to this set s of n so this set is set of all permutations of summation over all permutations of product. A i sigma of i and this i runs from 1 to n. So, sigma is basically the element where i is getting sent to. So, this is what the function permanent is.

And this is very similar to another function which all of you are more familiar with which is the determinant function and that has a similar definition except for one small change there is a sign factor so when you consider this products, sorry I am using M here, so there is a sign of the permutation that is considered as well. And the surprising thing is although this is just a small mathematical change between the two definitions.

But the complexity theoretic change that happens as a result of this is quite vast. So, permanent is actually hard for sharp p whereas determinant can be shown to be contained in p that is in Nc Nc 2 and the P algorithm is again quite simple it is basically the Gaussian elimination. So, if you apply Gaussian elimination on a matrix you only need to apply this polynomially many times to get a diagonal matrix.

And then once you have that it is just a product of the diagonals times whatever Gaussian elimination steps that you applied. So, that will give the polynomial time algorithm and but this is a much harder problem so we look at why this is a sharp p hard next class.