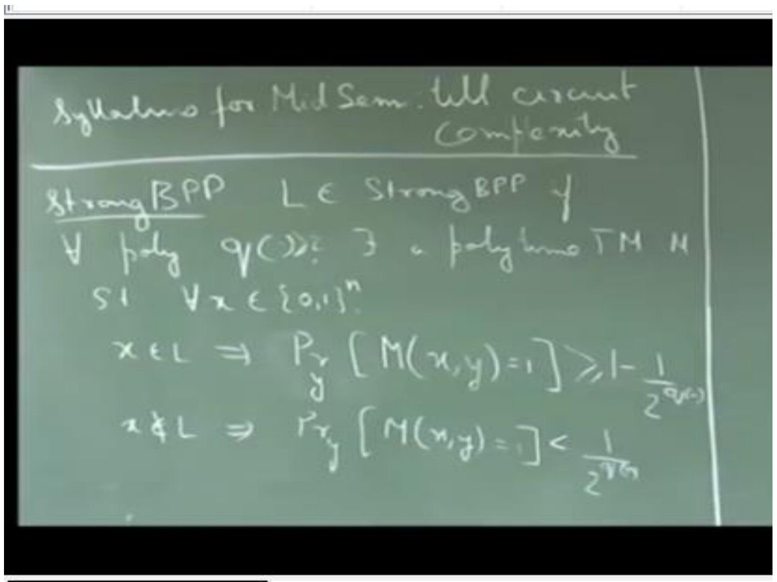
Computation Complexity Theory
Prof. Raghunath Tewari
Department of Computer Science and Engineering
Indian Institute of Technology-Kanpur

Lecture-19
StrongBPP and WeakBPP

So, let us get started. So, firstly I just want to make a correction. So, last class I said that the error probability has to be on either side of half. So, that is actually not necessary for BPP. So, you can have error probability that is on either side of half, but with some polynomial sized gap between the accepting and the rejecting probabilities and that still can be simulated by a BPP machine.

So, even the book does not give that proof, the book only proves it for a slightly weaker case when it is just polynomially bounded away from half on both sides, but we can actually prove a stronger theorem, but I would not prove that theorem today. So, I will prove it after mid sem, but anyway I will go ahead and state that what exactly I mean by these statements the definitions and what is the corresponding result. So, we will prove that. So, I was just trying to check it and I found that actually it is possible to do that.

(Refer Slide Time: 01:33)



Syllabus for Mid Sem. till circuit complexity

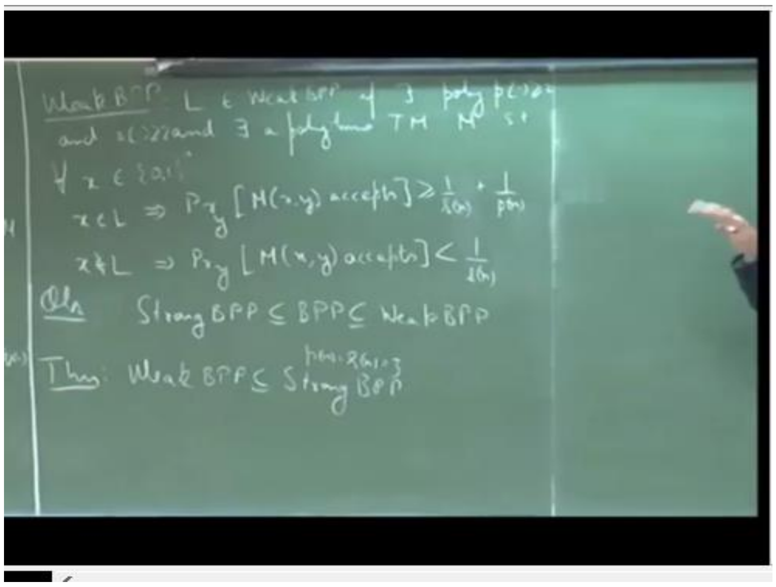
StrongBPP $L \in$ StrongBPP if
$\forall$ poly $q(\cdot)$: $\exists$ a poly time TM M
s.t. $\forall x \in \{0,1\}^n$
$x \in L \Rightarrow \Pr_y[M(x,y)=1] \geq 1 - \frac{1}{2^{q(n)}}$
$x \notin L \Rightarrow \Pr_y[M(x,y)=1] < \frac{1}{2^{q(n)}}$

So, let us define a something called strong BPP. So, L is said to be in strong BPP, if for all polynomial q, there exist a polynomial time turing machine M such that for all x of length n, x is in L implies that the probability that M given x, y where this y is chosen uniformly at random, accepts is greater than 1 - 1 over 2 to the power q n and if x does not belong to L probability over y that M given x, y.

Again accepts is less than 1 over 2 to the power q n. Let us also assume that this polynomial q is just greater than or equal to 2, because if q is the polynomial 1 that always outputs 1 then we have a problem with the definition. Then this will be greater than or equal to half and this is less than half. So, it does not create any gap. As long as q is any polynomial that gives a value larger than or equal to 2, this should be fine.

So, what this means is that I can take I mean I can assume the error probability to be exponentially close to 1 if with M outputting the correct answer. So, if I have an instance which is in the language then it outputs 1 with probability that is very close to 1 and if it is not in the language then also it outputs correct answer with probability very close to 1. Let us define weak BPP.

(Refer Slide Time: 04:40)



Weak BPP: $L \in$ Weak BPP if $\exists$ poly $p(\cdot) \geq$ [illegible] and $s(\cdot) \geq 2$ and $\exists$ a poly time TM M s.t.

$\forall x \in \{0,1\}^*$

$x \in L \Rightarrow \Pr_y[M(x,y) \text{ accepts}] \geq \frac{1}{s(n)} + \frac{1}{p(n)}$

$x \notin L \Rightarrow \Pr_y[M(x,y) \text{ accepts}] < \frac{1}{s(n)}$

Obs: Strong BPP $\subseteq$ BPP $\subseteq$ Weak BPP

Thm: Weak BPP $\subseteq$ Strong BPP

[illegible]

So, is this clear the definition. So, we say that L is in weak BPP, if there exist polynomials let me call them p and s and there exist a polynomial time turing machine M such that for all x of a certain length, x belongs to L implies that the probability that M x y accepts is greater than or equal to s n + 1 over p n and if x does not belong to L then the probability that M x y accepts is less than or equal to s n.

So, here basically the gap is some inverse polynomial and it does not matter what the base is, the base can be any polynomial. So, I should not write it this way, maybe I should write it as.

So, let me write it as 1 over s n and do I need s n to be greater than 2 here? Yes p n should be greater than 2. I think we can also enforce that s to be greater than 2 as well, does not matter.

But the point is that I can so basically I am free to choose a base value with a gap of some polynomial size. So, now the statement of the theorem is that strong BPP is contained in BPP which is contained in weak BPP. So, this is actually easy to prove, I will not state this as a theorem, this is probably just an observation, because if you take a language L in strong BPP for all polynomial this holds.

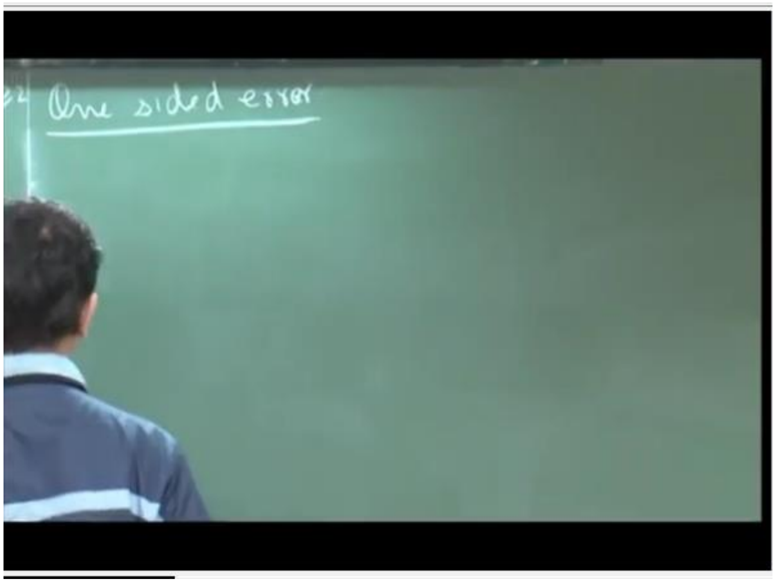
So, in particular I can fix this polynomial to be I mean if it holds for all polynomial which is greater than 2 it immediately implies that if x is in L then the probability is greater than two thirds and if x is not in L then the probability is less than one third, because this number can at most be one fourth. So, that immediately implies. You take a language L in BPP then again I mean you can fix s n to be. So, if you have a language L in BPP then it accepts with probability two thirds.

So, that will be two third and that will be less than one third. So, that is one choice, that is true. So, for this I can pick s n to be 3 and same as p n. So, basically what strong BPP saying is that so it has a more stricter condition of acceptance and rejection. So, it says that I only accept if it is very close to 1 and the probability needs to be very high. So, it has a much more stricter condition for acceptance as well as for rejection.

So, it demands the gap to be very large, whereas BPP that only demands the gap to be some constant between two third and one third, but I am also defining weak BPP where first of all the gap is only inverse polynomial and it does not matter around which point that gap is as long as it is some number between 0 and 1. For example if it is some constant between 0 and 1 that is still fine I mean I can choose the gap on either side of that constant.

So, the theorem is which I would not prove today is that weak BPP is contained in strong BPP. So, actually if you have something like this that probability of correctness can actually be amplified as large as this value. So, is the idea clear what is happening? So, let us so far we saw probabilistic turing machines which had a probability of making an error on either types of input that is on instances which can belong to L as well as on instances which does not belong to the language.

(Refer Slide Time: 12:20)



But now let us look at a different type of probabilistic machines once that only are allowed to make error on one side.