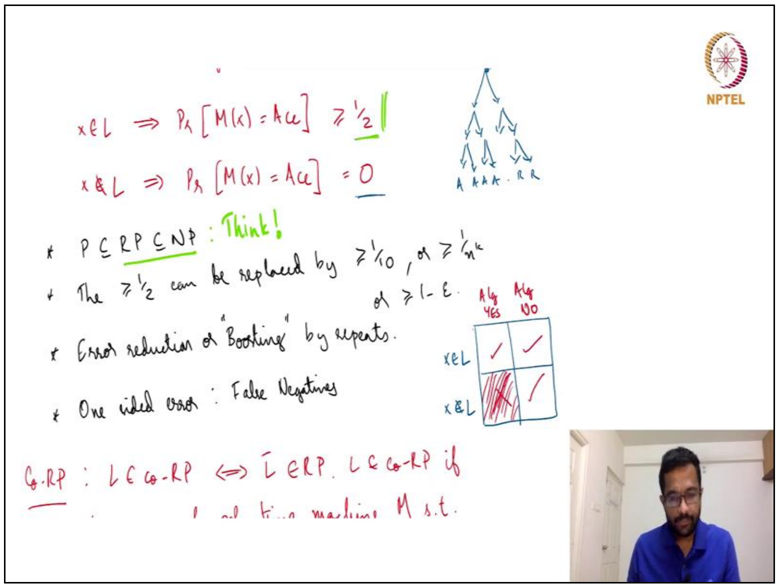
Computational Complexity
Prof. Subrahmanyam Kalyanasundaram
Department of Computer Science and Engineering
Indian Institute of Technology, Hyderabad

Lecture -28
Randomized Complexity Classes Part 2

(Refer Slide Time: 00:15)



$x \in L \Rightarrow \Pr[M(x) = Acc] \geq \underline{1/2}$

$x \notin L \Rightarrow \Pr[M(x) = Acc] = \underline{0}$

- $P \subseteq \underline{RP \subseteq NP}$: Think!
- The $\geq 1/2$ can be replaced by $\geq 1/10$, or $\geq 1/n^k$ or $\geq 1-\varepsilon$.
- Error reduction or "Boosting" by repeats.
- One sided error : False Negatives

	Alg YES	Alg NO
$x \in L$	✓	✓
$x \notin L$	▨	✓

$\underline{\text{Co-RP}}$: $L \in \text{co-RP} \Leftrightarrow \bar{L} \in RP$. $L \in \text{co-RP}$ if

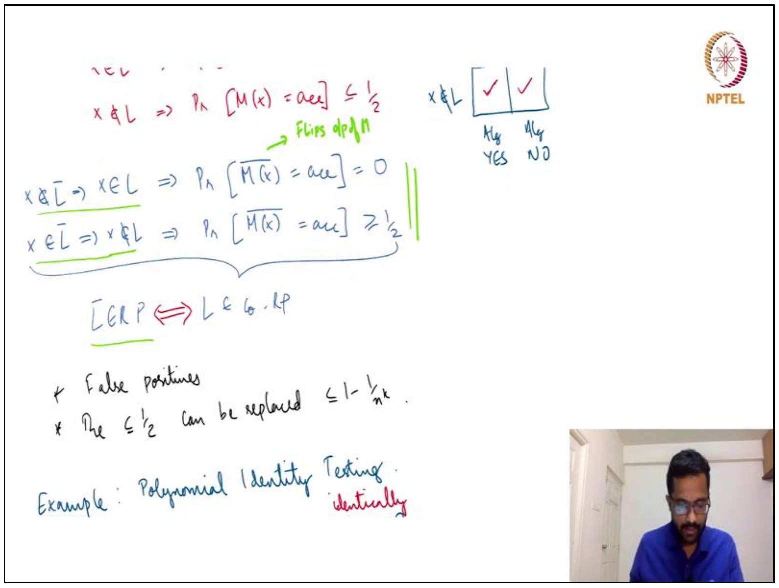
M s.t.

Hello and welcome to lecture 28 of the course computational complexity. In the previous lecture we motivated randomized algorithms and we explained we define the complexity class RP which is it stands for randomized polynomial time which is a complexity class used to describe computational problems or languages that have a one-sided polynomial time algorithm with one-sided error, so, the one-sided error being false negatives.

So, whenever there is an x that is in the language it may report that the algorithm may report that x is not in the language but not the other way around. So, today we will see in this lecture we will see 2 more complexity classes or perhaps 3 more one starting with co-RP. So, co-RP just like we have seen co-NP and co-NL stands for all the languages that are complement of the languages that are found in RP.

So, the first definition is along the same lines L is in co-RP if L complement is an RP. So, if and only if L complement is in RP. So, another way to define it is which is which is equivalent you will see why they are equivalent is that L is in co-RP if there is a probabilistic polynomial time machine probabilistic machine which runs in polynomial time such that whenever x is in the language it accepts with probability one and whenever x is not in the language it accepts the probability at most half.

(Refer Slide Time: 02:15)



$x \notin L \Rightarrow \Pr[M(x) = acc] \leq \frac{1}{2}$

Flips o/p of M

$x \notin \bar{L} \Rightarrow x \in L \Rightarrow \Pr[\overline{M(x)} = acc] = 0$

$x \in \bar{L} \Rightarrow x \notin L \Rightarrow \Pr[\overline{M(x)} = acc] \geq \frac{1}{2}$

$\bar{L} \in RP \Longleftrightarrow L \in co\text{-}RP$

$x \notin L$ | ✓ | ✓ |
Alg YES, Alg NO

* False positives
* The $\leq \frac{1}{2}$ can be replaced $\leq 1 - \frac{1}{n^k}$.

Example: Polynomial Identity Testing. identically

So, maybe over here we have the grid picture for RP. So, now let me draw the similar grid for co-RP. So here it is algorithm outputs yes algorithm outputs no x is in the language and x is not in the language. Whenever x is in the language it will accept always whenever x is not in the language it may it may accept but it could also of course it can reject but it may also accept. However this zone we never get into this zone meaning there is this only false positive in x and x that is not in the language may be incorrectly reported as being in the language but not the other way around.

So, in the case of RP we had only false negatives but in the case of co-RP we have false positives. So, this is the description of co-RP. Let us see why this definition is the same as the previous definition. Suppose there is an x. So, consider an L which has this description this meaning the description over here maybe I will just for the co-RP machine we will show why the complement of that language L has to be in RP.

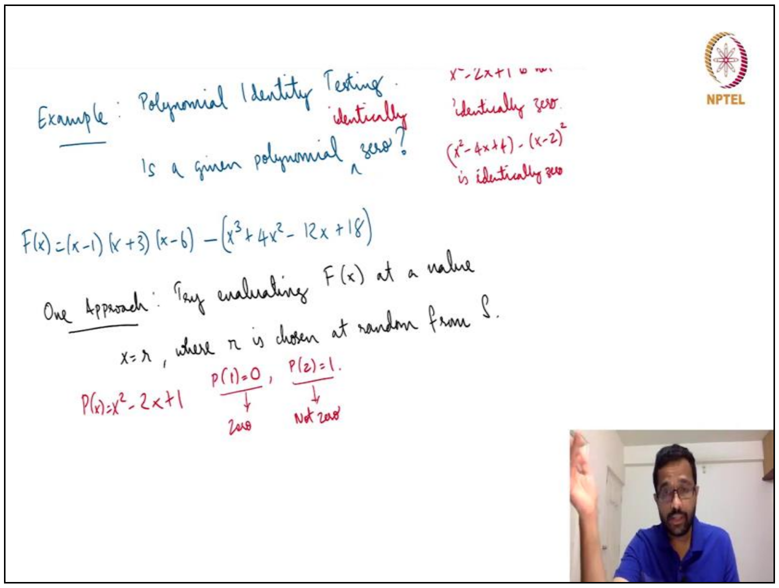
Suppose x is an L then it we can we can say that x is not an L complement that is the same thing and if x is not in L complement. So, I will raise this circle if x is not in L complement x is in L in that case the probability of M x accepting is with probability 1. So, now think of a machine M bar. So, M bar just flips the output of M flips output of M. So, if x is in the language then M the probability of M accepting is one. So, the probability of M bar accepting is zero because M bar is just going to say the opposite of whatever M is saying.

And whenever x is not in the language the probability of M bar accepting is at least half because all the yes's becomes accepts become rejects and the rejects become accents. So, the probability of M accepting was at most half the probability of M bar accepting was at least half but then x is not in L means that x is in L complement. So, now when we view this these 2 statements these 2 statements the ones that are written in blue as statements of with respect to L complement and the machine M bar we see that M bar is an RP decider for L complement.

So, whenever x is in L complement M bar accepts with probability at least half which is the RP definition and whenever x is not an L complement M bar accepts with probability zero. So, which is again the RP definition. So, hence we can conclude that L is in RP. So, we started with sorry L complement is in RP we started with L being in co-RP and we are able to infer that L complement is in RP. So, if and only if and similarly you can make the other direction inference as well.

So, this is what we saw now is one direction inference L is in co-RP implies L complement is an RP but the other direction is also not difficult to see. Basically whatever we just saw you can make the exact same arguments in the reverse direction.

(Refer Slide Time: 06:23)



Example: Polynomial Identity Testing.

Is a given polynomial $_\wedge$identically zero?

$x^2 - 2x + 1$ is not identically zero.

$(x^2 - 4x + 4) - (x-2)^2$ is identically zero

$F(x) = (x-1)(x+3)(x-6) - (x^3 + 4x^2 - 12x + 18)$

One Approach: Try evaluating $F(x)$ at a value $x = r$, where r is chosen at random from S.

$P(x) = x^2 - 2x + 1$ $\quad \underset{\text{zero}}{\underline{P(1) = 0}}$, $\quad \underset{\text{Not zero}}{\underline{P(2) = 1}}$.

Again some very quick points there could be false positives in a co-RP machine but no false negatives. And just like we could replace the half in the RP with an arbitrary quantity just like that this half here the half over here this half also can be replaced within uh. So, this is this shows up the acceptance for x naught in the language should be at most half but then this could be instead of this half we could have at most 2 thirds or 3 fourth or 0.9 and we can even come very close to one we could come polynomially close to 1 like it could be 1 - 1 over n power k.

But we cannot come closer than 1 - inverse polynomial this is what and still if you have 1 - 1 by n over k it could still be boosted and the probability of error can be reduced. So, 1 - 1 over n over k this means that the probability of error is sorry probability of error is sorry whenever x is not in the language this is a probability of error. So, but even this one - inverse polynomial error can be brought down with polynomial many repeats.

So, again the way the repeats work etcetera it is it is very very similar and I think I believe this notes has some I have formally written down this point but we can go through it when i share the notes. I will just explain one more example but in fact this is also a co-RP example sorry RP example but this is another problem. So, this problem is called polynomial identity testing. So, it may sound very simple but it happens to be a very fundamental problem which is which is very interesting especially in terms of in term in the in the perspective of randomized algorithms.

The problem is a given polynomial identically zero. So, what do I mean by identically zero. So, consider this x square - 2 x + 1 this polynomial can be set to 0 let us say when x is equal to 1 this problem this polynomial evaluates to 0 this evaluates to 0 but this is not always 0 when x is equal to 3 for instance you get 9 - 6 + 1 which is equal to 4. So, this polynomial is not is not identically zero sorry sorry there is some issue it is not identically zero.

However may be to give an example of another polynomial that is identically zero could be these are the polynomials that that are always zero maybe. So, maybe, so, x squared minus something like this x squared - 4x + 4 - x - 2 square. So, if you expand the second term x - 2 squared you will get exactly the same expression that is the first term and. So, they will cancel out. So, this is identically 0. So, it should always be equal to 0 no matter what value you assign to x that is what identically 0 means.

So, for instance I am here in the main text there is this polynomial f x. So, is f of x identically zero well. So, it is written as a difference. So, both the terms the leading term the leading monomial is x cubed. So, here if you multiply these 3 things x - 1 x + 3 and x - 6 we get x cube the constant term of the of the left hand side or the first term is - 1 into + 3 into - 6. So, it is +18 here also it is +18 but if you if you if you look closely for instance if you look at the x squared coefficient.

So, x squared coefficient is just the sum of the the 3 constants. So, it is 3 - 1 - 6 which is actually -4. So, x the term the coefficient of x squared is incorrect. So, we understand that this is actually not identically zero but when you look at it first glance it is not very clear so one. So, this is the problem polynomial identity testing. So, let us try for a simple algorithm simple randomized algorithm that can be used to decide this problem. So, one pro one approach is take the polynomial.

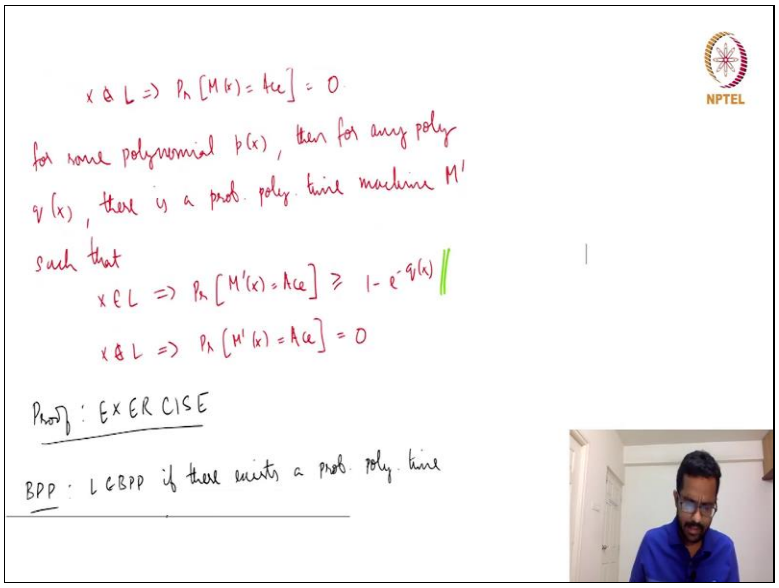
So, and you just substitute some value let us say x is equal to 5 and then see whether what this polynomial evaluates to. If it evaluates to zero then you say it is zero polynomial if it evaluates non-zero it says non-zero polynomial. So, let us see what is the success rate and what is the failure rate. So, let us take the earlier example x square - 2 x + 1. So, if you happen to evaluate this at x is equal to 1 you will report you will get 0 and then you will report that this is the 0 polynomial.

However if you evaluate it at x is equal to 2 for instance you will get you will evaluate the evaluation. So, x is equal to 1. So, let us say this is P x P of 1 is 0 whereas P of 2 is 1. So, if you evaluate it x is equal to 0 or sorry x is equal to 1 you get 0 and if this is the random point at which you decide to evaluate the algorithm will report the algorithm will report it is zero it is zero polynomial whereas this the algorithm will report not zero polynomial.

So here there are actually. So, I said wrong I earlier said this is a RP it has an RP algorithm but this is a co-RP algorithm because the polynomial could be not the zero polynomial but we may there are false positives we may incorrectly state that a non-zero polynomial is a zero polynomial. So, here it is certainly not a zero polynomial but we are reporting it as a zero polynomial. So, this is a false positives are there but notice that we will never have false negatives this is because false negatives mean we incorrectly report something as a as not a zero polynomial when it is a zero polynomial.

But when the actual polynomial is a zero polynomial no matter what you substitute it into you will get the evaluation to be zero.

(Refer Slide Time: 14:27)



$x \notin L \Rightarrow \Pr[M(x) = \text{Acc}] = 0$

for some polynomial $p(x)$, then for any poly $q(x)$, there is a prob. poly. time machine M' such that

$x \in L \Rightarrow \Pr[M'(x) = \text{Acc}] \geq 1 - e^{-q(x)}$

$x \notin L \Rightarrow \Pr[M'(x) = \text{Acc}] = 0$

Proof : EXERCISE

BPP : $L \in BPP$ if there exists a prob. poly. time

So, this is a we can get false positives but not false negatives. So, this is a co-RP algorithm it is a polynomial time algorithm you evaluate the polynomial at a certain value. However so, this seems very simple algorithm but this is a I said that this is a fundamental important problem this is a problem for which in the case of single weight. So, all these polynomials we considered have only one variable x but when we have multiple variables.

The multivariate case of this product this problem there is no efficient way of deciding whether a given polynomial is a 0 polynomial or not in deterministic time. But in the case of randomized algorithm we have this algorithm that I just described. Whatever I just described here you could also describe you could also generalize to multivariable multivariate case. So, you just pick x is equal to something and y equal to something all of this you can do.

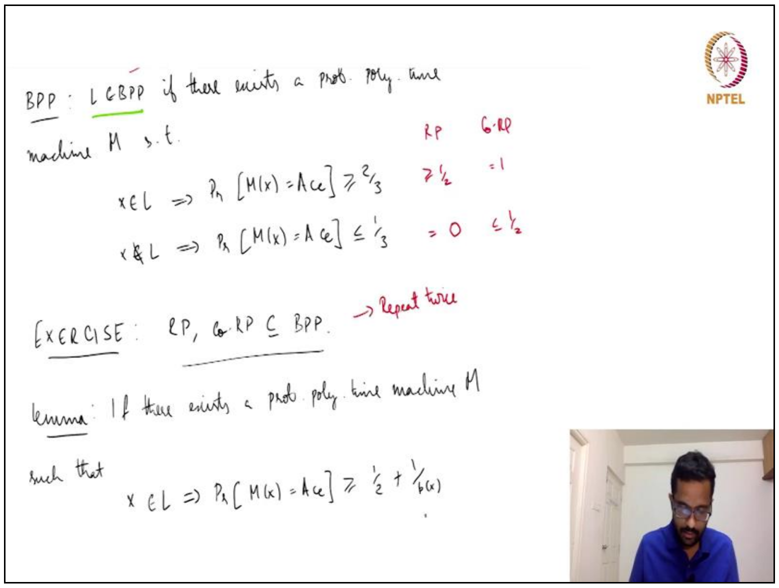
But there is no known deterministic algorithm. So, this is one this is a candidate problem that makes one thing that randomness does indeed gives more power as compared to deterministic polynomial time algorithms. So, this is a co-RP algorithm. So, let me just describe this boosting as an exercise. So, what I have stated here is that if we have an algorithm again this is this is stated in the RP setting.

If you have an algorithm for which the probability of acceptance is 1 by P x where P x is a polynomial. So, its inverse polynomial and probability that x naught in L is accepted is zero. So, this is RP setting then we could by doing enough boosting. So, here we have inverse polynomial something very very close to zero one by P x we could boost that to something like so something extremely close to 1 you could by repeating you could get 1 - e power - q x where q is any polynomial.

So, think about how to; so, there is nothing much to think about is rather working out it is exactly what I described. So, when you repeat it you will have to which answer do you say. So, when one is a yes and one is a no which answer do you say this one part that you have to think and then what is the probability of error? The probability of error is when x is not in when x is in L and it is still reported as a false negative it is reported as being not in L.

So, this probability will go down with repeats. So, just work out how many repeats are necessary to get from 1 by Px to 1 - e power - qx. So, this is that is the only part that that needs to be worked out but you can work out for and convince yourself. So, that concludes the small part about co-RP now I want to move to a 2-sided error model.

(Refer Slide Time: 18:16)



BPP : $L \in BPP$ if there exists a prob. poly. time machine M s.t.

RP Co-RP

$x \in L \Rightarrow P_r[M(x) = Acc] \geq 2/3$ $\geq 1/2$ $= 1$

$x \notin L \Rightarrow P_r[M(x) = Acc] \leq 1/3$ $= 0$ $\leq 1/2$

EXERCISE : RP, Co-RP $\subseteq$ BPP. $\rightarrow$ Repeat twice

lemma: If there exists a prob. poly. time machine M such that

$x \in L \Rightarrow P_r[M(x) = Acc] \geq \frac{1}{2} + \frac{1}{p(x)}$

So, till now we have seen RP and co-RP both cases we had one-sided error RP had false negatives but no false positives. Co-RP had false positives but no false negatives now we are going to see BPP which stands for Bounded Error Probabilistic Polynomial time this is this is shortened as BPP. BPP, L is in BPP if there is a probabilistic polynomial time machine such that whenever x is in the language it is accepted with probability 2 thirds and whenever x is not in the language it is accepted with probability at most one thirds.

So, in the case of RP we had the first one was greater than half and second one was equal to zero. In the case of co-RP the first one was one equal to one and the second one was less than or equal to half. So, here instead so, at in both the cases one of them was fixed at the desired value but in the case of BPP both of them both of the errors are possible. So, the false negatives are possible false positives are both are also possible.

However the probability of acceptance in both the cases is different or the bounds are different. So, the bounds are different. So, the key thing here to note is that between 2 thirds and one thirds there is a gap. So, it is not like when something is in the language the probability of acceptance is significantly higher than when it is not in the language. So, this is at least 2-thirds it is at most one-thirds and yeah so, I will first state this too.

First thing is that RP and co-RP are contained in BPP this is a small exercise. So, again as I said sorry as I said if we had an RP machine this was at least half and this is equal to zero if you had a co-RP machine this was equal to 1 and this was at most half. So, if we have an RP machine then how do we get a BPP machine? So, this is this does not fit in the BPP model because the probability of when x is in L the probability of acceptance is only at least half.

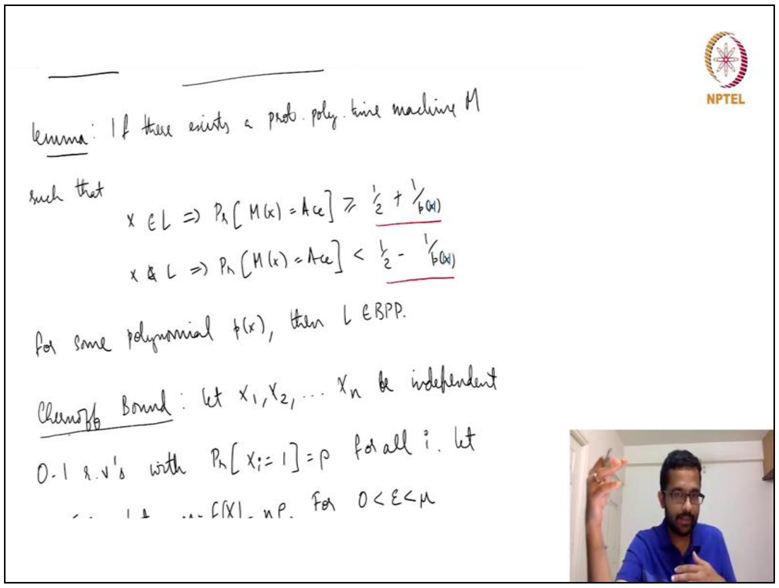
When x is not in L the probability of acceptance is zero which is because BPP requires it to be at most one third and this is at most one third. What we can do is to repeat it twice what would be the error for an RP when you repeat it 2 times? The error here is at most half but the error will go down to one fourth because both the; trials have to be successful or both the trials have to be incorrect.

So, this half will become 3-4ths which is bigger than 2-3rds and similarly in the case of co-RP. So, just by repeating the RP algorithm 2 times we will get the probability of success or property of acceptance in the case of x is equal to L to be at least 3-fourths. And the probability of acceptance in the case of x is not equal to L remains zero. So, again repeat twice and then when both the trials are both one of the trials accept in one of the trials is reject.

Then we know that there are possibility of false negatives. So, we say accept because whenever it says accept we know for sure it is correct in the case of RP. In the case of co-RP the decision is the opposite whenever you accept only when all of them accept because there are false positives but then the same argument will take down the probability of error to one-fourths. So, when x is not in L the probability that it the probability that it is accepted can be upper bounded by one fourth.

So, you can work out this if necessary we could work out this in a in one of those meetings as well live sessions as well.

(Refer Slide Time: 23:36)



Lemma: If there exists a prob. poly. time machine M

such that

$$x \in L \Rightarrow \Pr[M(x) = Acc] \geq \underline{\tfrac{1}{2} + \tfrac{1}{p(x)}}$$

$$x \notin L \Rightarrow \Pr[M(x) = Acc] < \underline{\tfrac{1}{2} - \tfrac{1}{p(x)}}$$

for some polynomial $p(x)$, then $L \in BPP$.

Chernoff Bound: let $X_1, X_2, \ldots X_n$ be independent

0-1 r.v's with $\Pr[X_i = 1] = p$ for all i. let

... $E[X] = np$. For $0 < \varepsilon < \mu$

And just like we saw for RP and co-RP where we mentioned that this half is arbitrary even the 2 thirds and one thirds in the case of BPP are arbitrary. But let us see to what extent we have we can move them around. So, one thing that you may guess is that if I make the probability of if I make both of them from 2 thirds if I bring down to half and this one third is also brought up to half then it is there is no distinction between x being in the language and x being not in the language both of them both of them will accept with probability half.

So, then you may actually just output accept and reject with probability half without looking at the input. So, that is not going to be useful. So, we need some demarcation between x is in the language and x is not in the language the probability of acceptance when x is not in the language should be something above half and the property of acceptance when x is not in the language should be something just below half.

So, how much gap should we maintain between half or between the x's in the line x is in the language and x is not in the language. So, it turns out that inverse polynomial is enough. So, if we have the probability of x if there is a probability probabilistic polynomial time machine such that x is whenever x is in the language it is accepted with probability half plus inverse polynomial and whenever x is not in the language it is accepted with probability at most half minus inverse polynomial.

So, the only gap that we are saying is half plus inverse polynomial in the case of x is in the language and half minus inverse polynomial that is the upper bound when x is not in the language for any polynomial P x. So, in fact it is strictly speaking it is P size of x because x is the input and the length of the input is what we are referring to then L is in BPP meaning then we can. So, what this really says is that if we have L which or if you have a machine M for L which satisfies this half + 1 by P x and half - 1 by P x then by then by boosting we can get to 2 thirds one thirds.

Let us see how we can do that in the case of one sided error the boosting was easy because we only got one type of error. So, when we saw SAT is like when we saw the when we had a guess a random assignment and say that phi is satisfiable not satisfiable. If we had even one of the trials outputting phi is satisfiable the trials outputted satisfiable because we found a satisfying assignment. So, then we did not need to look at we can we could safely ignore all the other trials that outputted phi is not satisfiable.

However so, what I am saying is that the boosting in the case of one sided error was relatively easy. However in this case both could happen x being in the language being incorrectly recorded as x P not in the language and x being not in the language being incorrectly reported as x being in the language false negatives as well as false positives. So, we cannot afford to do this kind of decision making.

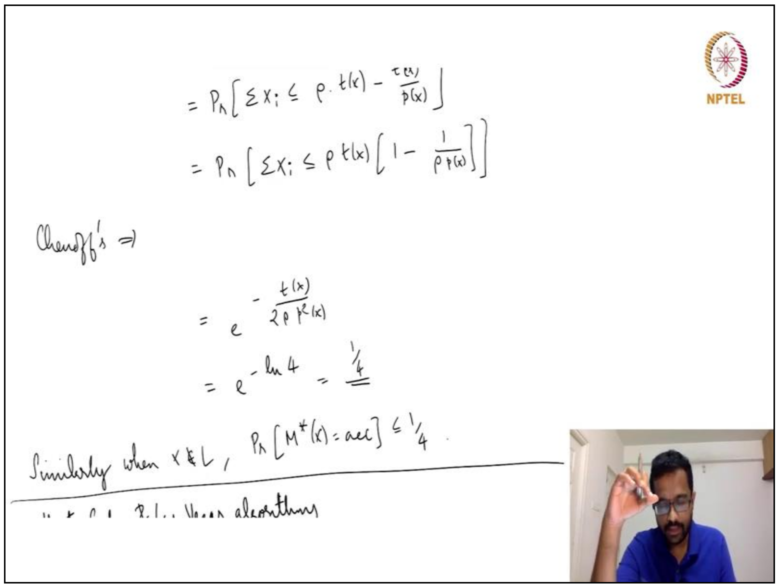
If there is at least one accept does not mean that you can accept same for reject. So, we need to do something else. So, what we will do I will first explain what we do and then explain why we do this? What we do is we run the machine some number of times let us say t number of times we repeated t number of times and then we decide based on the majority. So, we decide let us say we run 100 times and if 51 of the times it is accept.

We accept and let us say anything else you reject and the claim is that when the claim is that when x is when if we have even this kind of gap x is in the language half + 1 by P x, x is not in the language the probability of acceptance is at most half - 1 by P x then if you repeat it many times then it is likely that the number of expected number of accepts is significantly higher than expected number of rejects.

So, let us say this is true x is in the language half + 1 by P x not in the language half - 1 by P x then let is say you repeat thousand times again this is just to give a intuition then the probability of accepts are likely to be around let us at least 505 and the probability of rejects are going to be at most 495. So, because there is a gap and this gap is pronounced when you repeat it in one trial it is it is not pronounced it is very close to half but when you repeat it many times this gap becomes pronounced.

So, when you repeat it you are say you are likely to see more than 505 accept if x is in the language or less than 495 rejects when x is oh sorry 495 access when x is not in the language. So, you can safely accept or reject.

(Refer Slide Time: 29:34)



$$= P_r\left[\sum X_i \le \rho \cdot t(x) - \frac{t(x)}{p(x)}\right]$$

$$= P_r\left[\sum X_i \le \rho\, t(x)\left[1 - \frac{1}{\rho\, p(x)}\right]\right]$$

Chernoff's $\Rightarrow$

$$= e^{-\frac{t(x)}{2\rho\, p^2(x)}}$$

$$= e^{-\ln 4} = \underline{\underline{1/4}}$$

Similarly when $x \notin L$, $P_r[M^*(x) = acc] \le 1/4$.

[illegible] algorithms

Based on this is the key idea for this however I will not go into the details of this because it is a bit technical. This is executed by something called Chernoff bound. What is Chernoff bound it says that if there are n random variables x 1 to x n n n 01 random variables and all of them being independent and all of them being identically distributed with probability rho meaning the probability that each one of them being equal to 1 is rho and the probability of each one of them being equal to 0 is they are 01 random variables.

So, probability that x, x i equal to 0 is 1 - rho and this is true for all the i's all the x's. Now consider x to be summation of x i just how many of them are 1 and the expectation of x is mu and expectation of x is nothing but n times rho because it is for each one of them the expectation is rho n times the expectation is by linearity of expectations it is n times rho. And in this case Chernoff bound says that. So, maybe just look at these 2 statements over here these 2 red statements it says that the probability that x is bigger significantly bigger than the mean.

So, mu is the mean of expectation of x the probability of x being significantly bigger than the expectation is e power minus epsilon squared by 3 times mu and it being significantly smaller than the expectation is also again not that significantly it is e power minus epsilon square. So, the point is that this exponential quantity. So, what it actually says that if x is this is the mean of x. So, suppose this is a mean of x mu if we have done sufficiently many trials the expected many trials are done then we are likely to see x being very closely concentrated around the mean.

So, this is x the distribution of x, x is very likely to be strongly concentrated around the mean it is unlikely to vary significantly the probability of x moving significantly away from you in the positive or the negative side is very very low. So, the probability of x going far away is very small that is what Chernoff bound says. And so, when you repeat it many times this gap of half + 1 by P x and 1 - P x becomes more and more pronounced.

And the rest is just calculating this. So, the there is a machine we call it we repeat M t times. So, again we have to evaluate what is t what kind of t works and this machine this this machine that or this repeated machines. Let us call it M star this accepts if at least half of the outputs are accept otherwise it rejects and then we can calculate the probability of error and thanks to Chernoff bound this probability of error is 1 4th for either case for x being in the language and sorry when x is not in the language sorry if x is in the language what is the probability that x is not accepted.

This comes out to be one-fourth. So, this which means that this probability when x is in the; language it will be accepted with probability at least 3-fourths and the below one when x is not in the language it will be acceptable with property at most one-fourth which is better than 2-thirds one one-thirds that the BPP definition suggests. So, this is what i want to say about BPP again once again i want to say that this 2-thirds one-thirds could be replaced by 3-fourths one-fourths or 9 by 10, 1 by 10 or as we saw here it is it could be very close to half also.

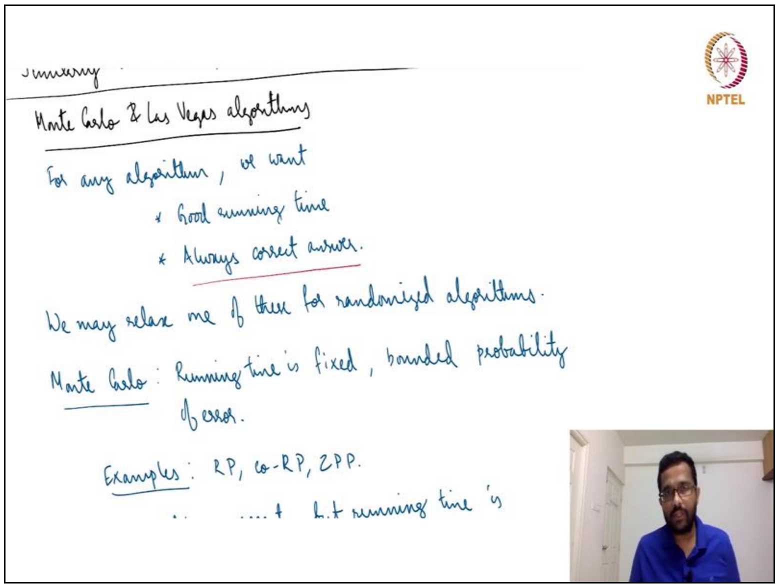
Half plus some constant half minus not half plus constant or certainly works but half + 1 by polynomial also works and half - 1 by polynomial. So, but just by repeating it many times and by virtue of Chernoff bound we could we can infer that even from a small from a small gap half + 1 by P x and half - 1 by px you can boost it sufficiently to get something like 2-thirds one-thirds or 3-fourths one-fourths.

So, all that you do is you repeat it many times and whatever is the majority outcome appears outcome that happens more number of times you output that and rest of the rest; so, that the calculation even though it may seem like many symbols and calculations it is just 2 things one is just applying Chernoff bound and 2 you figuring out what is the number of times that one has to repeat, so, that we will get a constant. So, the number of times one has to repeat is rho times 2 times P square into ln 4.

So, the number of repeats again just polynomial that is why it was important to have a polynomial 1 by P x here if instead of 1 by P x if we had one by 2 power x or something then this would not have worked because then in place we would have to repeat exponentially many times. So, that is about BBP and how we can boost the probability of success in BPP. However as I already said BPP contains both RP and co-RP.

And usually when somebody says a randomized algorithm one tends to think of BPP because it is it is more general than RP and co-RP. This RP and co-RP are restricted in terms of the error type allowed one of them allows only false negative one of them only false positive but BPP is more versatile that way. So, and it contains both of them. So, it is easier to work with BPP assumption.

(Refer Slide Time: 36:42)



Summary

Monte Carlo & Las Vegas algorithms

For any algorithm, we want

* Good running time
* Always correct answer.

We may relax one of these for randomized algorithms.

Monte Carlo: Running time is fixed, bounded probability of error.

Examples: RP, co-RP, ZPP.

[illegible] ....t b.t running time is

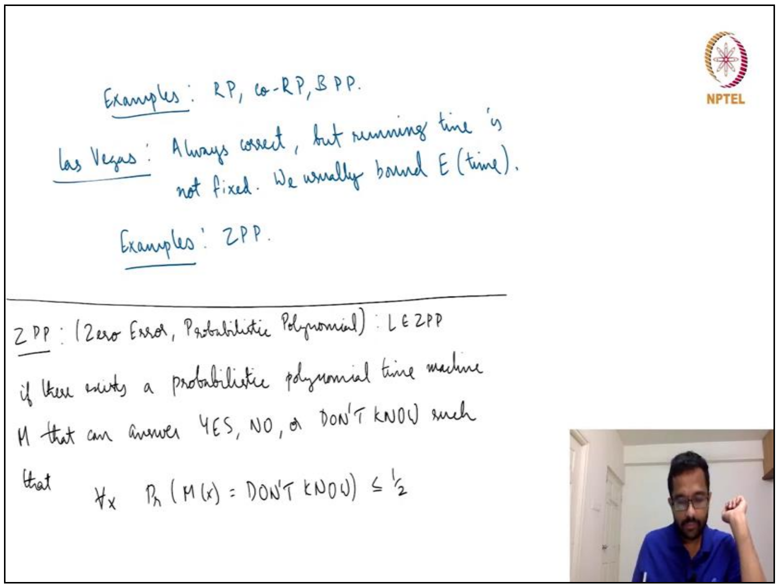
And I will conclude this lecture by saying to to one point about what is called Monte Carlo and Las Vegas algorithms. So, in any algorithm these are 2 different Monte Carlo and Las Vegas are 2 cities one of them is in Europe Monte Carlo is in Europe somewhere near Italy and Las Vegas is in USA in the state of Nevada both of them are known for like very flashy lifestyle and presence of Casinos.

So, people tend to gamble over there which is why they have been associated with the class of randomized algorithms and randomized complexity classes. So, in any algorithm what do we want in any deterministic algorithm say we want the outcome to be correct and we want to happen efficiently in good running time 2 things correct output in decent time and in the case of randomized algorithms because things are random meaning the setting is random you make random choices.

So, we relax we tend to relax one of these. So, there are 2 things that we could relax. So, till now what we have seen is we have been relaxing the requirement of always correct answer. This requirement we have been relaxing how have we been relaxing this by allowing for some error probability we allow some error but then we try to make the error control. So, that we can manage the probability of error in this end and the running time is fixed.

So, one situation is when we relax the correctness requirement but keep the running time requirement unchanged and this is called Monte Carlo algorithm running time is fixed but the probability of error we try to manage you keep it as a fixed quantity. So, that you could or you upper bounded lower becomes something. Whatever we have seen so, far RP co-RP and BPP are examples of complexity classes that pertain to Monte Carlo algorithms.

(Refer Slide Time: 39:09)



Examples: RP, co-RP, BPP.

Las Vegas: Always correct, but running time is not fixed. We usually bound $E(\text{time})$.

Examples: ZPP.

ZPP: (Zero Error, Probabilistic Polynomial): $L \in ZPP$ if there exists a probabilistic polynomial time machine M that can answer YES, NO, or DON'T KNOW such that $\forall x \quad \Pr(M(x) = \text{DON'T KNOW}) \leq \frac{1}{2}$

What we have not seen so, far is Las Vegas algorithms where we relax the running time requirement. So, there the answer will always be correct but the running time may change. So, the running time may increase decrease based on the input. So, one example even though it is not a decision problem is quick sort. We know you may have heard of this randomized quick sort the randomized quick sort chooses the pivot randomly.

And you may have studied that even in the case of randomized quick sort even when the pivot is randomly chosen. The running time could be as bad as n squared the worst case running then could be as bad as n squared however it could perform well and in fact in the case of randomized quick sort most of the time it is expected to perform well. And the running time is usually number of comparisons expected number of comparisons is n log n.

So, the running time varies but the correctness is always. So, at the end of it whether you take n log n time or n square time the array that is given to you is sorted correctly. So, that the correctness is always guaranteed. This is the class of Las Vegas algorithm running time fluctuates but algorithm is always correct. Sometimes what we do in the case of quick sort we say expected running time is bounded. So, we bound expected running time.

And the complexity class that pertains to this one complexity class that pertains this is called ZPP. ZPP stands for zero error probabilistic polynomial time that is abbreviated as ZPP and I think we will see this in the next lecture and I will quickly summarize. What we have seen so, far we have seen co-RP which corresponds to the languages that are the complements of the language that are the complements of the languages in RP.

And we saw that it has false positives but no false negatives there also we saw the probability of success that half is an arbitrary choice and we have some leeway there. We saw polynomial identity testing which and saw a co-RP algorithm for it where we potentially can get false positives it could even though the polynomial is not identically 0 we may incorrectly declare it as identically 0 but it cannot have false negatives.

Then we saw BPP which is bounded error probabilistic polynomial time sorry and this allowed for both errors it is not one-sided it is 2-sided error. So, this is called 2-sided error both false negatives and false positives it allowed where the only thing that we really need is a gap between the acceptance probability in the case when x is in L and x is not in L even though we write 2 thirds one thirds it could be 3 4ths 1 4th or 9-10ths 1-10ths and we saw that this the gap could be brought as close as half plus inverse polynomial and half minus inverse polynomial just leaving a inverse polynomial gap.

And this how we can boost it we boost it by using Charnoff bound we repeat it many times and we take the majority outcome. And I did not get into the details of the Charnoff bound calculation but it is rather standard in the field of randomized algorithms and I urge you to look that up. Finally we saw what is Monte Carlo and Las Vegas algorithms. Monte Carlo algorithms are those where running time is fixed but correctness is changes because of some errors are possible. So, we bound errors.

Las Vegas algorithms are running time is running time varies but are always correct. So, we bound the like expectation of running time or something like that. And all that we saw that we have seen so, far RP co-RP and BPP correspond to Monte Carlo algorithms and what we are yet to see is a complexity class that corresponding to that corresponds to Las Vegas algorithms called ZPP which stands for zero error probabilistic polynomial time which we will see in the next lecture.