

**Computation Complexity Theory**  
**Prof. Subrahmanyam Kalyanasundaram**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology-Hyderabad**

**Lecture-30**  
**Randomized Complexity Classes: Part-4**

(Refer Slide Time: 00:58)

PP: PP consists of those languages L for which there  
 is a probabilistic polynomial time TM M such that

$x \in L \Rightarrow P_x [M(x) = Acc] \geq \frac{1}{2}$   $\geq \frac{2}{3}$   
 $x \notin L \Rightarrow P_x [M(x) = Acc] < \frac{1}{2}$   $\leq \frac{1}{3}$

$x \in L \Rightarrow P_x [M(x) = Acc] \geq \frac{3}{4}$   
 $x \notin L \Rightarrow P_x [M(x) = Acc] < \frac{3}{4}$

\*  $RP \subseteq PP$   
 + The value  $\frac{1}{2}$  is an arbitrary choice.  
 + Boosting?  
 $NP \subseteq PP$

Hello and welcome to week 6 and lecture 30 of the course computational complexity. So, in the last week we saw randomized complexity classes RP, co-RP, ZPP and BPP. Today in this lecture we will see a new class called PP which stands for probabilistic polynomial time and then we will elaborate a bit on how BPP is boosted, how the probability of success is boosted in the case of BPP.

And will contrast between BPP and PP and we will also compare all the randomized classes towards the end of this lecture. So, what is PP? PP stands for probabilistic polynomial time, so it is a very simplistic name, it does not give you any clue what it is except that it is a probabilistic class. So, basically these are the class of languages for which there is a probabilistic polynomial time turing machine just like what we had for RP, co-RP, BPP and so on.

But the when x is in the language the probability of acceptance is just above half or just greater than or equal to half and when x is less than or x is not in the language the probability of acceptance is strictly less than half. So, this is how PP is defined. So, contrast this with

BPP, so BPP the way we defined it I think it was two-thirds and if  $x$  is not in the language this was less than one-third. So, we could just go back and refer back to our notes two thirds and one thirds.

(Refer Slide Time: 02:15)

$x \in L \Rightarrow P_x [M(x) = \text{Acc}] \geq \frac{2}{3}$   
 $x \notin L \Rightarrow P_x [M(x) = \text{Acc}] \leq \frac{1}{3}$

Proof: EXERCISE  
 Bounded Error Probabilistic Polynomial Time  
 BPP:  $L \in \text{BPP}$  if there exists a prob. poly time machine  $M$  s.t.

two sided  
 $x \in L \Rightarrow P_x [M(x) = \text{Acc}] \geq \frac{2}{3}$   
 $x \notin L \Rightarrow P_x [M(x) = \text{Acc}] \leq \frac{1}{3}$

EXERCISE:  $\text{RP}, \text{co-RP} \subseteq \text{BPP}$ .  $\rightarrow$  Repeat twice  
 and: poly time machine

So, this is what we used. So, in the case of BPP it is two-thirds and one-thirds. In the case of PP it is greater than or equal to half and strictly less than half. So, notice what happens here. In the case of PP there is really no gap, so if exactly with half probability a certain input is accepted and even if one of the random strings decides to flip. So, suppose there are all the randomness that is used are of 10 bits long which means there are 2 power 10 1024 possible random bit strings. If exactly half of them 512 lead to accept and 502 lead to reject  $x$  would be in the language in the case of if it is a PP machine.

Even if one of them becomes reject from accept, so then it will be 511 accept and 513 reject. In that case it would be  $x$  not in the language. So, it is a very fragile situation there, whereas in the case of BPP the gap there is a significant gap when  $x$  is not in the language when is  $x$  is in the language it has to be accepted with probability at least two thirds and when  $x$  is not in the language it is at most one third. I mentioned that this two thirds and one thirds are arbitrary and that we could increase or reduce the gap but still at even then there will always be a gap.

(Refer Slide Time: 03:56)

Lemma: If machine  $M$  has the property that

such that

$$x \in L \Rightarrow P_x[H(x) = \text{Acc}] \geq \frac{1}{2} + \frac{1}{\text{poly}(n)}$$

$$x \notin L \Rightarrow P_x[H(x) = \text{Acc}] < \frac{1}{2} - \frac{1}{\text{poly}(n)}$$

For some polynomial  $f(n)$ , then  $L \in \text{BPP}$ .

Chernoff Bound: let  $X_1, X_2, \dots, X_n$  be independent 0-1 r.v.'s with  $P[X_i = 1] = p$  for all  $i$ . let  $X = \sum X_i$ . let  $\mu = E[X] = np$ . For  $0 < \epsilon < \mu$

So, the boosting we said that it can be boosted in such a way that the probability for accept  $x$  is in the language is half +  $\frac{1}{\text{poly}(n)}$  by  $P$  and  $x$  is not in the language is half -  $\frac{1}{\text{poly}(n)}$  by  $P$ . So, still there is an inverse polynomial gap between accept and reject, whereas in the case of PP there is absolutely no gap. So, this is a difference between the 2 classes PP and BPP. So, the first thing to note is that RP is contained in PP because RP just requires when  $x$  is in the language the probability of acceptance to be at least half.

When  $x$  is not in the language the property of acceptance should be 0. So, an RP machine by definition is also a decider for PP. Because 0 is less than half and another thing to note is that the value half is an arbitrary choice. So, instead if I had a machine that accepts you may if you look at other references you may see some things like this  $x$  is in the language then probability of acceptance is strictly greater or three-fourths for instance and when  $x$  is not in the language probability of acceptance is less than three-fourths.

So, when I say arbitrary you could move the half bar to anywhere like three-fourths or one-fourths or whatever. It is not very difficult to see why this can be done but even when the bar is moved the accept and reject remain very close to the bar. So, there is no gap between accept and reject. So, even if one random bit flips from one accept to the reject or reject to accept the output flips.

So, just that where the bar is that can be moved. Whereas in the case of BPP it is like a there is a gap and of course the gap can also be shifted a bit but there will always be a gap. So, even if you bring it closer it will still not be the same, there will always be a gap in the case

of BPP. However in the case of PP the gap is not there, but where we want the threshold between accept and non-accept or  $x$  being in the language and not being in the language this can be moved.

The point one way to see this is you could just simply declare half the random bits to be leading to an acceptance. So, that only the rest of the half are actually going to determine the outcome of the string. So, then automatically you have half and if among the rest of the half leads to accept, half leads to reject then you have a situation like three fourths is the bar for acceptance.

So, that is a very high level idea but that is how we can move the bar from half to three fourths or maybe to nine tenths or maybe to one fourth. So, you could just make the similar kinds of argument and boosting. So, whenever I mentioned earlier classes we said about boosting in fact I will also elaborate the BPP boosting a bit in this lecture. However, there is no known boosting situation in the case of PP.

This is because there is no gap, so there is no way to repeat and kind of gain more information because the situation is very close. So, there is no boosting available and another point to note is that NP is contained in PP.

**(Refer Slide Time: 08:05)**

NP ⊆ PP

- \* For language  $L \subseteq NP$ , let  $V$  be the verifier.
- \* let  $x \in L$  have  $y$  as the certificate.
- \* For all strings of length  $|x|$ , let certificate length be  $p(|x|)$ .
- \* Consider randomized TM  $M$ , that takes random bit string  $r$  of length  $p(|x|)+1$ .

Diagram:  $x \rightarrow M \rightarrow 0/1$ , with  $r$  (random bit string) as input to  $M$ .

So, if there is one certificate, so NP what does it say? If  $L$  is an NP, then if  $x$  is in  $L$  then there is at least one accepting string, one proof or witness or certificate, but if  $x$  is not in the language then there is no such certificate that lead to an acceptance. So, you have a verifier

machine that will accept or not accept these certificates. So, we can basically convert the verifier or use build on the verifier as a randomized polynomial time machine to show that  $L$  is in  $PP$  as well.

So, how do we do that? So, for the language  $L$  that is in  $NP$  let  $V$  be the verifier. Now suppose if  $x$  is in the language  $L$  that  $y$  be the certificate. So, if  $x$  is in  $L$  then there will be a certificate let us call it  $y$ . Now if an input has a certain length let us say  $I$  am denoting  $x$  with vertical bars around it to denote the length of the string  $x$ . Let the certificate length be  $P$  of  $x$  where  $P$  is some polynomial.

So, by the definition of  $NP$  the certificate length has to be polynomial in the length of the input. So, let the certificate length be  $P$  of  $x$ . Now so  $V$  is a verifier and  $P$  is the length of the certificate,  $P$  of the length of the input. Now consider a randomized turing machine  $M$ , so which takes us input  $x$  as well as a random string as input. So, randomized turing machine meaning it just takes as input as some randomness/.

After that it just does some calculation based on the input as well as the random string. So, it is just like this. So, there will be  $M$ , there will be the input  $x$  and there will be some random string  $r$  and based on this there will be the output. This is how randomized turing machines works. So, and what  $M$  does is completely deterministic once it has the random bits?

So, the randomness comes is input through the random bits  $r$ . So, now consider randomized turing machine  $M$  that takes a bit string  $r$  as input where the length of  $r$  is one more than the length of the verifier. We will see why that is required, why it has to be one more than the length of the verifier? So, right as of now  $M$  is a machine that we are constructing. We will see how we use  $V$  in the construction of  $M$ . So, what does  $M$  do?

**(Refer Slide Time: 11:05)**

→ If  $r = 0u$ , then let  $M(x, r) = M(x, 0u) = V(x, u)$

→ If  $r = 1u \rightarrow$  If  $u = 111\dots 1$ , then  $M(x, r) = \text{reject}$   
 If  $u \neq 111\dots 1$ , then  $M(x, r) = \text{acc}$

→ Bounded Error Probabilistic Polynomial Time.

So, if  $r$  is of the form or the leading bit of  $r$  is 0, let us say  $r$  is 0 followed by some string  $u$ , where  $u$  is of length  $P(x)$ , so you will automatically be of length  $P(x) + 1$ . Then let so what is  $M(x, r)$  or  $M(x, r)$  which is equal to  $M(x, 0u)$ , we will define it to be  $V(x, u)$ . So, the first bit is 0 then we just run the verifier on the rest of it. So, whatever the verifier does?

If  $r = 1u$  then  $r = 1u$  there are 2 sub cases, if  $u = 1, 1, 1$ , all ones then reject if  $u$  is not equal to all 1s then so when I say reject I meant  $M(x, r)$  equal to reject, otherwise this is equal to accept. So, basically for all the inputs which start with for all  $r$  that start with 1 we accept except for a single string which is all one string. Let us see why we do that.

So, let us try to understand, so what is happening here. So, amongst all the bit strings that start with 1 everything leads to an acceptance except one string which is all one string. So, let us compute the probability of a string getting accepted.

**(Refer Slide Time: 13:53)**


NPTEL

If  $x \neq 111 \dots 1$ , then  $M(x, 1) = 0$

What's the prob of  $M$  accepting  $x$ ?

$$\frac{2^{P(x)} - 1 + |\text{Certificates}|}{2^{P(x)+1}}$$

BPP: LBPP  $\rightarrow$  Bounded Error Probabilistic Polynomial Time.  
 if there exists a prob. poly. time  
 $\therefore M$  s.t.



So, let us compute the probability, what is the probability of  $M$  accepting  $x$ ; there are 2 cases; one if  $x$  is in  $L$  so in other words another way to see this is how many random bit strings lead to accepting  $x$ ? So, let us say when  $x$  is in  $L$ , so how many random bit strings lead to  $M$  accepting  $x$ . So, for all the random bit strings that start with 1 all of them accept except 1. So, there are total  $2^{\text{power } P \times \text{many}}$  such strings all of them accept -1.

This is just the strings that begin with 1 and plus number of certificates. So, all of them accept  $2^{\text{power } P \times} - 1$  and then  $r$  begins with 0, all the correct certificates that lead to  $x$  being accepted. So, divided by total number of random bit strings which is  $2^{\text{power } P \times + 1}$ , so where 1 here is in the exponent. So, in fact this is actually true for everything not just  $x$  is in  $L$ .

**(Refer Slide Time: 15:48)**

NPTEL


If  $x \neq 111 \dots 1$ , then  $M(x, 1) = 0$

What's the prob of  $M$  accepting  $x$ ?

$$P(\text{acc}) = \frac{2^{P(x)} - 1 + |\text{Certificates}|}{2^{P(x)+1}}$$

When  $x \in L$ ,  $|\text{Certificates}| \geq 1$ . So  $\text{Prob}(\text{acc}) \geq \frac{1}{2}$   
 $x \notin L$ ,  $|\text{Cert}| = 0$ . So  $\text{Prob}(\text{acc}) < \frac{1}{2}$

$\rightarrow$  Bounded Error Probabilistic Polynomial Time.



But when  $x$  is in  $L$ , so this is the probability of acceptance, when  $x$  is in  $L$  what happens is there is at least one certificate, number of certificates is at least 1. So, probability of acceptance is at least, so this 1 and this 1 will at least cancel, so then you have  $2^{\text{power } P x}$  divided by  $2^{\text{power } P x + 1}$  which means this is at least half and when  $x$  is not in  $L$  number of certificates is actually equal to 0.

There is no certificates that lead to a string getting accepted, so probability of certificates or probability of acceptance will be just this entire term will go, the certificate thing will vanish, this is  $2^{\text{power } P x - 1}$  divided by  $2^{\text{power } P x + 1}$ . So, this is just below half mark so this is strictly less than half, so which means it is a PP machine that we have constructed. When  $x$  is in the language the probability of acceptance is strictly is greater than or equal to half and when  $x$  is not in the language the probability of acceptance is strictly less than half.

So, it is a PP machine. So, we took a arbitrary NP language and we showed that it is in PP. So, NP is contained in PP. So, this is about PP and NP containing in PP and it is not very difficult to see that BPP is also contained in PP, why is that? This is because as I said at the beginning this is the BPP numbers. So, any BPP machine automatically qualifies to be a PP machine.

The probability of acceptance is at least  $\frac{2}{3}$  at most one-thirds, at least two-thirds means it is all obviously above one-half, at most one third is all obviously strictly less than one half. So, a BPP machine is already a PP machine. So, BPP is also contained in PP.

**(Refer Slide Time: 18:16)**





It is not very difficult to see, so now I want to go on to the boosting part of BPP. Again last lecture I mentioned it briefly but I want to elaborate it a bit more. So, because it is more relevant to what is going to come after this. So, this is just recalling the definition of BPP, this is what I wrote in the previous lecture  $L$  is in BPP, there is a probabilistic polynomial machine such that that probability of acceptance at least two-thirds when  $x$  is in the language and at most two-thirds.

At most one-third if  $x$  is not in the language and the boosting result is that if there is a polynomial time machine that uses randomness such that if it accepts with probability  $\frac{1}{2} + \frac{1}{2^p}$ , so  $\frac{1}{2^p}$  by inverse polynomial could be some small number, it could be 0.001 or something. Again a constant is better than inverse polynomial. And when  $x$  is not in the language it is the probability of acceptance is upper bounded by  $\frac{1}{2} - \frac{1}{2^p}$ .

So, there is a cap but a smaller gap not two thirds one thirds it is just like  $\frac{1}{2} + \frac{1}{2^p}$  above half and  $\frac{1}{2} - \frac{1}{2^p}$  below half which is a  $\frac{2}{2^p}$  cap. Even this cap is can be boosted to a bigger one that is what we want to show. So, even if this is the situation for some polynomial  $P$  then that language is said to be in PPP. So, why is that language in BPP because we can boost the probability of success in this machine to get the two-thirds one-thirds one?

So, we will see how to boost the probability of error or reduce the probability of error or improve the property of success in order to get to the two-thirds one. So, here you can see the error is like  $\frac{1}{2} - \frac{1}{2^p}$  excerpt even when it is something is in the language just above half gets accepted. So, the probability of successes just above half, probability of error is just below half.

And when  $x$  is not in the language again it is the probability of acceptance is actually probability of error. So, that is just below half. So, how can this be boosted? What we use is a standard kind of technique from probability theory and also very much use in randomized algorithms called Chernoff bound. So, these are  $n$  independent 0, 1 random variable, they are independent that they have the same distribution.

Each one of them equal to 1 with probability  $\rho$   $x_1, x_2$  up to  $x_n$  and each one of them being 0 with probability  $1 - \rho$  obviously because they can only be 0 or 1. So, if they are not 1 they have to be 0. Now let us consider the sum of all of  $x_1, x_2$  up to  $x_n$ , the sum is let us

say  $x$  and  $\mu$  be the expectation of the sum and expectation of the sum or a set of bunch of random variables is nothing but the individual expectations.

But the individual expectation of each of them is  $\rho$  because  $0$  into  $1 - 0 + 1$  into  $\rho$  is just simply  $\rho$ . So, expectation of  $x_1$  is  $\rho$ ,  $x_2$  is  $\rho$  and so on. So, expectation of  $x$  is  $n$  times  $\rho$  which is  $n\rho$ . In that case what is Chernoff bound say?

**(Refer Slide Time: 22:13)**

$$P_n[X \leq \mu - \epsilon] \leq e^{-\frac{\epsilon^2}{2n}}$$
 For all  $0 \leq \epsilon \leq 1$ :
 
$$P_n[X \geq (1+\epsilon)\mu] \leq e^{-\frac{\epsilon^2}{3}\mu}$$

$$P_n[X \leq (1-\epsilon)\mu] \leq e^{-\frac{\epsilon^2}{2}\mu}$$
 Run  $H$  on input  $x$  a total of  $t(x) = p.l.f(n).ln 4$  times with independent random choices.
   
 $\rightarrow$   $\therefore$   $H$  fraction of accepts  $> 1/2$ .

Chernoff bound says that the probability the two forms of Chernoff bound the I am just stating both here that epsilon less than the expectation so again  $\mu$  is the expectation of  $x$  is  $e^{-\text{power epsilon square divided by } 2n}$ , that is one form. So, the main thing is that again this there are multiple forms of the Chernoff bound and if you look at wikipedia or other sources you may find different, different forms. They are all powerful, all of them are powerful and you may end up there is a chance that you get confused between them, many people do that.

So, do not be worried about that. The key thing to note is that the probability of  $x$  even epsilon being slightly away from the mean, even epsilon away from the mean, even that is exponentially small. So, the point is that this epsilon squared by  $2n$  appears in the exponent of  $e$ . So, as long as it is slightly away from the mean it just drops off significantly. That is the key thing that one has to focus.

So, like I drew it in this on the previous lecture, so if  $\mu$  is the mean then the  $x$  will be something like this, so it will be very closely concentrated around the mean. Even if you go slightly far away with the probability that it is going slightly far away drops down suddenly,

that is a key point here and the form that we will use is I think they are the same is the next one but which is identical to this one.

So, another form of stating this is for all epsilon between 0 and 1. So, earlier epsilon was in the previous form epsilon is a fixed quantity, but now epsilon is a 0, it is something between 0 and 1. So, now what is the probability that  $x$  is  $1 + \epsilon$  above  $\mu$  and  $x$  is less than  $1 - \epsilon$  times  $\mu$ . Again there are two forms here  $e$  when the first one what is the probability of  $x$  is  $1 + \epsilon$  above  $\mu$ , it is  $e^{-\epsilon^2}$  divided by  $3$  times  $\mu$  and so.

And the probability that  $x$  is at most  $1 - \epsilon$  times  $\mu$  is  $e^{-\epsilon^2}$  by  $2$  times  $\mu$ . So, the point what that one has to note is the expectation of  $x$  is  $\mu$ , what Chernoff bound is saying is what is the probability that  $x$  is  $1 + \epsilon$  times more than the expectation and this happens to drop off significantly. So, the point here is that when  $n$  is large when you, so this you may you can do a simple experiment and kind of convince yourself.

Let us say you toss a coin 10 times, we usually expect heads and tails to come roughly half, half of the times, but you may get let us say sometimes you may get 6 heads and 4 tails and that is not really that surprising, you can just toss it and try for yourself. Sometimes you make it even though it is less likely you may get 3 heads and 7 tails, but it is unlikely that all 10 of them are heads.

But now let us say you repeat the experiment let us say 1000 times, let us say 10000 times, if you toss a coin 10000 times it is very, very unlikely that you will see 6000 heads and 4000 tails, what you will see is that most of the times at least 5490 heads and between 4900 and 5100 you will see both heads and tails will be between 4900 and 5100, it is not going to be like vary that much.

So, the point is that when you repeat these experiments more and more when  $n$  becomes more then mean is going to be more and more concentrated around the expectation or the sum is going to be more and more concentrated on the expectation which is what is happening here.

**(Refer Slide Time: 27:19)**

$$P_n [x \geq (1+\epsilon) \mu] \leq e^{-\epsilon^2 M}$$

$$P_n [x \leq (1-\epsilon) \mu] \leq e^{-\epsilon^2 M}$$

Run  $M$  on input  $x$  a total of  $t(x) = \frac{1}{\epsilon^2} \frac{1}{\rho(x)}$  times with independent random choices.

$M^*$  accepts iff the fraction of accepts  $> \frac{1}{2}$ .

let  $\rho = P_n [x_i = \text{acc}] \geq \frac{1}{2} + \frac{1}{2}\epsilon$

$$P_n [M^*(x) \neq \text{acc}] \leq P_n \left[ \sum_{i=1}^{t(x)} x_i \leq \frac{1}{2} t(x) \right]$$

NPTEL

So, now what we do is again what let us come back to what we set out to do, we had a machine that accepted  $x$  that is in the language with property one half +  $\epsilon$  and for  $x$  is not in the language the property of acceptance was at most half -  $\epsilon$ , what we do is we will exploit the Chernoff bound. How do we do that by repeating this machine multiple times like with any other boosting, we just run, we repeat this machine multiple times?

And then we see the majority outcome. So, if the majority is accept we will accept, if the majority is reject we will reject. So, the number of times that we will repeat is this quantity  $\frac{1}{\epsilon^2} \frac{1}{\rho(x)}$  times. So, why we get this number we will see. So, basically we will do a bunch of calculations and at the end if you put this quantity it just works out to be the correct quantity.

So,  $M^*$  is the machine that where we repeat  $M$  this many times on the same input  $x$  and we set  $M^*$  to accept if the fraction of accept is greater than half. So, now we set  $\epsilon$  again, what is  $\epsilon$  here,  $\epsilon$  is the probability that  $x_i$  accepts and this we know to be at least half +  $\epsilon$ .

**(Refer Slide Time: 29:03)**

NPTEL

$$= P_n \left[ \sum x_i \leq \rho t(x) - \frac{\epsilon t(x)}{p(x)} \right]$$

$$= P_n \left[ \sum x_i \leq \rho t(x) \left[ 1 - \frac{\epsilon}{\rho t(x)} \right] \right]$$

Chernoff's  $\Rightarrow$

$$= e^{-\frac{\epsilon^2}{2 + \epsilon t(x)}}$$

$$= e^{-\ln 4} = \frac{1}{4}$$

Similarly when  $x \leq L$ ,  $P_n [M^*(t) = acc] \leq \frac{1}{4}$

Now what is the probability that M star accepts? M star accepts when at least half of the trials accept. So, the  $x_i$  will be 1 if that the  $i$ th trial is accept, so what is the probability that M star is not accept, M star is not accept when strictly less than half the trials do not accept or maybe less than or equal like maybe half it is not strictly less than. So, less than or equal to half trials do not accept or in other words at most half the trials accept.

So, the number of accepted trials is less than or equal to half the total number of trials, so half is the  $T x$  is the number of tribes right and  $x_i$  indicates each trial whether it accepted or not.  $X_i$  is 1 when a trial accepts and 0 otherwise. So, again the next line I am dropping in this line the second line I am dropping the summation limits  $i = 1$  to  $T x$  and I am writing it in a different way.

So, I am writing half as so half  $T x$  I am writing it as  $\rho - 1$  by  $P x$  because from this quantity we have half plus 1 by  $p x$  to be less than or equal to  $\rho$ . By rearranging we get that half is less than or equal to  $\rho - 1$  by  $P x$ . So, this is what we have here, so we replace this quantity and then we can take the  $\rho T x$  outside by adding another  $\rho$  here, another  $\rho$  in the denominator here and now we have this form.

So, notice that  $\rho T x$  is the expectation, this is the expectation. So, now if you apply Chernoff this form what is the probability that  $x$  is less than or equal to  $1 - \epsilon$  times the expectation. So,  $\epsilon$  here is  $1$  by  $\rho$  times  $P x$  which is nothing but  $e$  power  $-\epsilon$  square by 2 multiplied by the expectation.

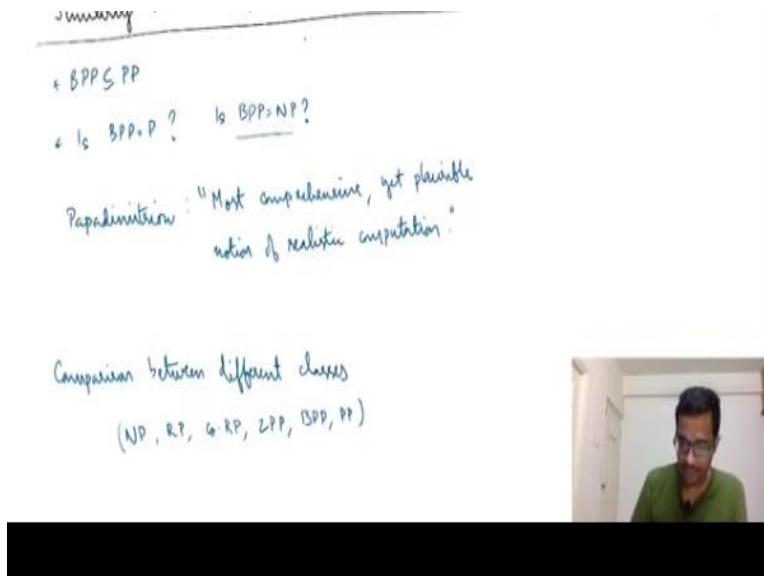
So,  $e^{-1}$  by so you can work that out it is  $\rho^2 P^2$  and then multiplied by the expectation which is  $\rho T^x$ , so the one  $\rho$  cancels and division divided by 2 and you have this quantity the denominator 2 times this and if you set  $T^x$  to be this quantity it is exactly this denominator multiplied by natural logarithm of 4, all that remains is minus natural logarithm of 4. So, it is  $1/4$ .

So, what we showed here is that when  $x$  is in the language the probability of this repeated the boosting machine not accepting is at most  $1/4$  and same similar calculation you can do when  $x$  is not in the language and you will again get this one fourth probability. So, basically we are repeating and then showing that the resulting machine; again we use only polynomially many repeats,  $P$  is a polynomial.

The number of trials is  $\rho \times 2 \times P^2 \times \log 4$ . So,  $\rho \times 2 \times \log 4$  are constants  $P$  is a polynomial, so  $P^2$  is also polynomial, so the number of repeats is still polynomial. So,  $M$  is a polynomial time,  $M^*$  is also a polynomial time machine because running time is just multiplied by a polynomial and the probability of accept when  $x$  is in the language is at least three fourths and probability of accept when  $x$  is not in the languages at most one-fourth.

Hence which conforms to the requirements of the BPP machine. Hence even if we had the probability of acceptance just above half and just below half we could boost it to get a PPP machine. So, again this is a standard kind of boosting but I thought it will be good to go through because it is good to know this and especially if you are interested in randomized algorithms and study you will find this very helpful. So, that is about the boosting of PPP. Again I already said that BPP is contained in PP.

**(Refer Slide Time: 34:16)**



And another point is that one question that one may ask is BPP equal to P it is widely believed that it is equal to P but there is no proof known. In fact it is not even known if it is equal to NP even this is not known, is it contained in NP even that is not known, whether BPP is contained in NP, this is not known. So, there are many open questions. So, widely believe that BPP is equal to P, but there is no proof yet.

And it is so whenever people think of randomized algorithms people I think the most natural class is BPP because it is natural to have 2 sided errors and some error probabilities in both cases rather than let us say one side error only one kind of error etcetera. So, in fact Papadimitriou says this in his book, it is the most comprehensive yet plausible notion of realistic computation.

So, notice that he does not say realistic randomized computation, it just says realistic computation. So, hence it is a very interesting class, it is a very fundamental class.

**(Refer Slide Time: 36:08)**

Papadimitriou: "Most comprehensive, yet plausible notion of realistic computation."

Comparison between different classes  
(NP, RP, G-RP, ZPP, BPP, PP)

And the last thing I wanted to say today in this lecture is about comparison between different randomized classes and I will start with NP just to get some baseline going. Let me just check on the time, so perhaps I will just do this in the next lecture. So, I will just summarize this lecture we define PP the fact that it does not have a gap, it does not have a gap.

We showed that NP is contained in PP, we saw BPP is contained in PP and then we saw the boosting of BPP and using Chernoff bound and I mentioned that it is a very important technique that is used a lot in randomized algorithms and in the next lecture we will see the comparison between different randomized classes.