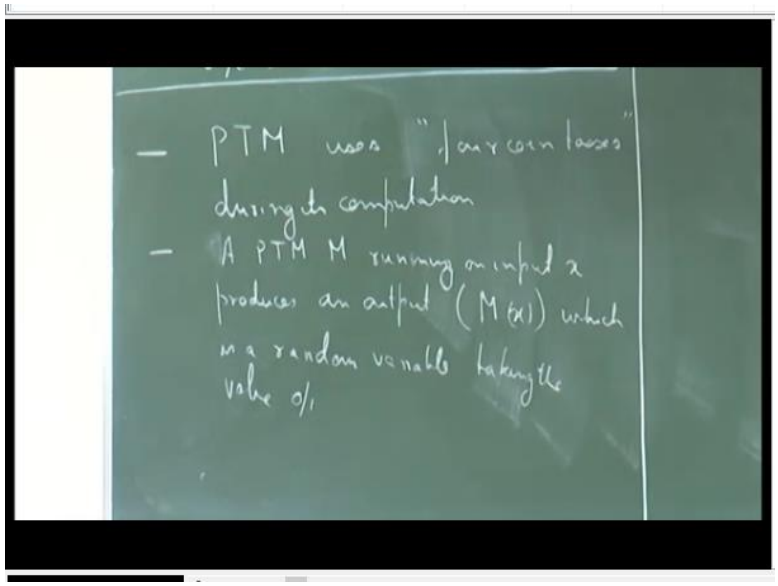
Computation Complexity Theory
Prof. Raghunath Tewari
Department of Computer Science and Engineering
Indian Institute of Technology-Kanpur

Lecture-18
Probabilistic Complexity

So let us continue. So, in the last few lectures, we talked about circuit complexity. In particular, we saw families of circuits having polynomial size and what are their computational powers. We also saw restrictions of circuit where we put some restriction on the depth of the circuit as well as on the fanin of that circuit. So, these gave us some interesting classes and we saw certain problems that belong to these classes.

So, what we will look in the next couple of lectures is randomized turing machines, or probabilistic turing machines. So, we will define what we mean by probabilistic turing machines and then we will analyze their computational power. In particular, we will look at what I mean analogous to how we define complexity classes for deterministic and non deterministic machines. We will define complexity classes for probabilistic turing machines as well and then we will study the relations between them.

(Refer Slide Time: 01:42)



So, just to get a quick idea as to what a probabilistic turing machine is and how it differs from the kind of machines that we have seen so far. So, a probabilistic turing machine, in short, we will denote it as PTM. It uses fair coin tosses during its computation. So, basically, at each step, together with what the current state is and what are the symbols that are being read by

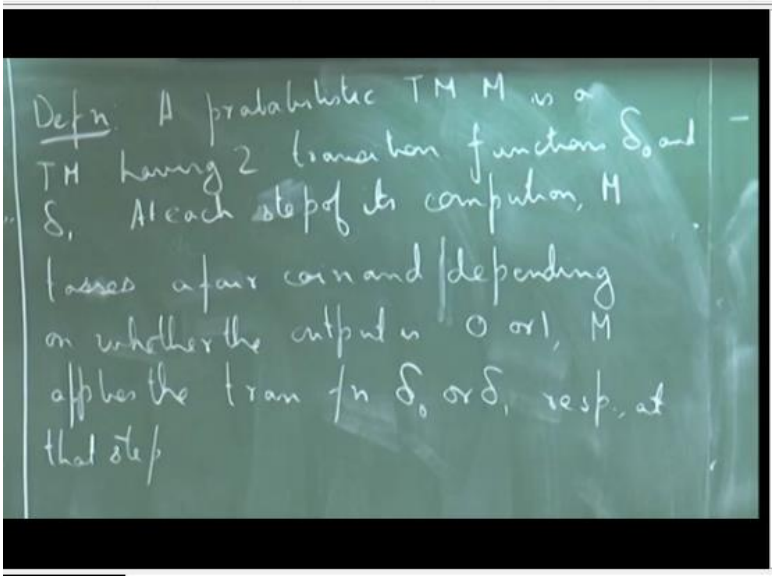
the input head, the turing machine also tosses will assume it is a fair coin, that is it produces a 0 1 output with equal probability.

And then depending on what the answer to that coin toss is, it will proceed accordingly. So, it is quite similar to a non deterministic machine in the sense that at each step you are deciding between 2 possible choices, but the difference actually, as we shall see, will arise in how the turing machine decides to accept its input. So, in a non deterministic machine, the machine accepts if there is at least one path, but here we will see that it has a quite a different accepting criteria.

So, since it has these coin tosses, what we can say is a probabilistic turing machine M running on let us see an input x produces an output. So, we will denote that by M of x, which is a random variable, taking the value, 0 or 1. So, now you can think of the output of a machine on an input as a random variable. And the reason why we call it a random variable because the machine on this particular input has a sequence of random steps.

So, whatever is the final output that it is producing that is a random very land it takes value, 0 or 1. So, this is the basic intuition. So, let us formalize the definition. And let us look at the acceptance criterion, which is the most important thing.

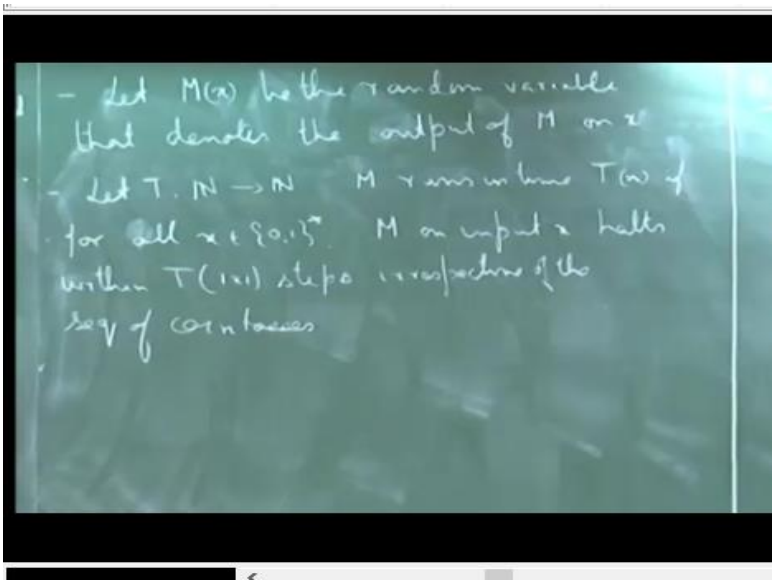
(Refer Slide Time: 05:20)



So, a probabilistic turing machine M is a turing machine having 2 transition functions, it is called delta 0 and delta 1 and at each step of its computation M tosses a fair coin and depending on whether the output is 0 or 1 M applies the transition function delta 0 or delta 1

respectively at that step. So, this is how, that is one way to think of it, but if you look at this definition, I will just think of that as that transition functions have the same definition at that step. So, both delta 0 and delta 1 do the same thing. So, it does not matter whether you get a 0 or a 1, you are taking that same step.

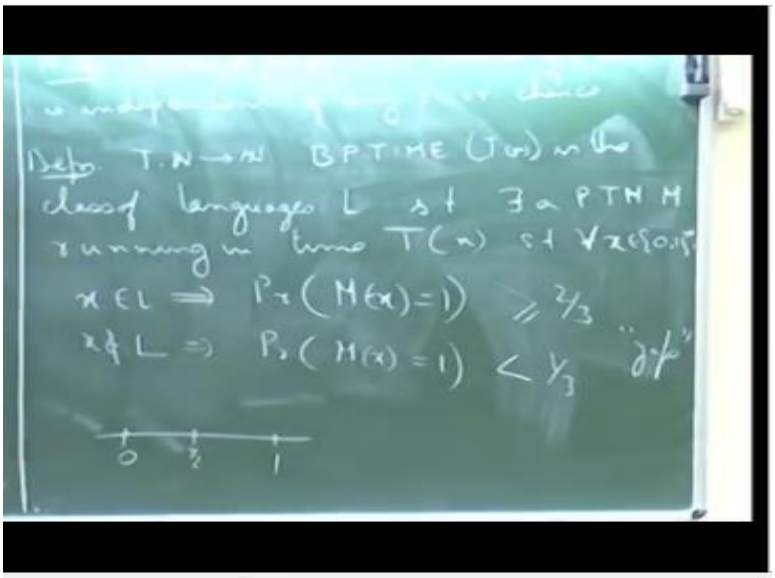
(Refer Slide Time: 07:57)



So, far a probabilistic turing machine M. So, let M of x the random variable that denotes the output of M on x. So, let us talk about running time. So, let T be some function from natural number to natural numbers. So, then we say that a probabilistic turing machine or I am already using that so, we say that M runs in time T of M, if for all x 0 1 start M on input x halts within T of x step irrespective of the sequence of coin toss.

So, no matter what sequence it gets, the machine on any input should halt within T of x number of steps. So, that is when we say that M runs in time T of M. So, far, I have not talked about the acceptance criteria, I have just talked about a general probabilistic turing machine and how it operates. So, we will have actually separate accepting criterions that we will consider. So, that is why I did not talk about it the definition of the machine.

(Refer Slide Time: 10:59)



So, just let us note one important characteristic of these machines. So, at each step during the running of this machine, so one thing is that it makes a random choice, depending on what that coin tosses and not only is it random, it is also independent of the prior choices that the machine has made up till that step. So, these 2 are very important. So, it does not I mean, it is action at any particular step.

If you look at the definition, it does not depend on what action it had taken prior to that step M choice of delta 0 or delta 1 is independent of any prior choice. So, now let us look at a class of languages that were defined for this probabilistic turing machines. So, again, let T be some function from natural numbers to natural numbers, then B P TIME P of n is the class have languages accepted by probabilistic turing machine M.

Let me just change this a little bit the class of language is L such that there exists a probabilistic turing machine M running in time T of n such that for all x. If x belongs to n, then the probability that the random variable M of x takes the value 1 that is that M on input x accepts is greater than two thirds and if x does not belong to L again the probability that M of x accepts is strictly less than one third.

So, in other words, there is a gap between the probabilities for an instance that is in the language and an instance that is not in the language. So, only if such a machine exists for a particular language we say that that language will belong to B P TIME of T n for some T of n. So, B stands for bounded. So, basically the point here is that it is actually another thing is that why do we arbitrarily choose this constant two third.

So, there is nothing magical about two third, it is just some constant that is bounded away from half. So, B stands for the term bounded and what we mean by bounded in this case is some number that is bounded away from half. So, if you look at the number line, so, between 0 and 1. So, these are the probabilities that by which M of x accepts or rejects and we have half.

So, now, we choose some constant that is bounded away from half and then we say that if an x belongs to the language then that machine on that input should output 1 with probability that is greater than this whatever constant this is and if it does not belong to the language then the probability should be less than this constant. And as we shall see that not only can we choose any constant that is greater than half actually we can choose any polynomial that is bounded away from half, 1 by 9.

No, no of course, we have to choose something that is on the other side of half because that is necessary at least for the case of B P TIME it is necessary that when we are accepting it should be with probability greater than half and not only it should be with probability greater than half it should be greater by some inverse polynomial term that is bounded away from half.

So, it can be something like half plus 1 over n to the power 10 or half plus some 1 over n to the power C where C is a constant, but it cannot be yes. So, that is the definition because if you look at the problem that we had in quiz, this class P P. So, there what is the gap between a string that is getting accepted and that is not getting accepted? Not exactly 0 there is some gap, it is basically 1 over some exponential term.

Because, if you look at that total I mean it is basically 1 over the total number of paths that are possible. So, the total number of paths that are possible for a polynomial time machine is 2 to the power n. So, what we said in that definition was that if a string was in the language I mean greater than half the number of accepting I mean paths were getting accepted and if it was not it was the other way around.

So, there was basically a gap of something like 1 over 2 to the power n. So, here by definition we do not allow such small gap. I mean, not exactly less than or equal to half. In the

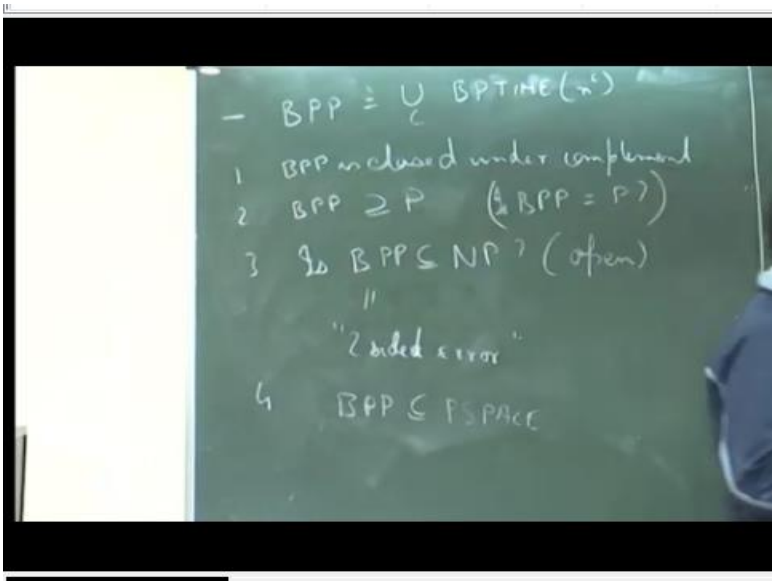
construction it came out to be exactly equal to half. So, that is why I am saying that this is different from that definition. So, they are by construction. I mean, not by construction, they are by definition it implied that the gap could be as close as 1 over 2 to the power n.

But here the way we define this class the gap has to be at least some 1 over n to the power C. No, by choice we do not include half on any of the 2 sides, half should be excluded. No, because I mean, you can look at this in a different way. So, we say that x does not belong to L, if the probability that M of x output 0 is greater than two thirds, I mean is greater than or equal to two thirds. So, that would look symmetric.

So, that is why we have less than one third, not necessary, that is not necessary, I mean, you can have arbitrary numbers here are also, you can have let us say, three fourth here and one third here, that is also fine. Only thing that is necessary is that this gap should be some inverse polynomial. And the same with this, I mean, at least some inverse polynomial and this also should be at least some inverse polynomial.

So, that is the only thing that we will see. So, we have not seen that so far. I mean, this is how we are defining it as of now, but we will see that some inverse polynomial will suffice. So, maybe tomorrow or maybe after next sem, I mean, not tomorrow, day after tomorrow. So, we have not talked about polynomial times yet. I mean, we are just talking about some time function T of n. So let us hold on for a moment..

(Refer Slide Time: 22:05)



So, now we can look at polynomial time machines and that gives us the class BPP. It has the same usual definition. So, it is union over all C of the classes BP TIME n to the power C. So, we look at probabilistic turing machines that run in polynomial time for some polynomial and we look at the languages that are accepted by them. So, now, what can we say about BPP. So, let us focus on this class.

So, the first thing is that if you look at the definition, you can immediately conclude that BPP is closed under complement because if you take a language which lies in OBPP. It basically means that it is for those strings that are not in the language, they are getting accepted by probability greater than two thirds and those things which are in the language they are getting accepted by probability less than one third.

So, what you can do is, you can just switch the answers, you can just flip the answers at the end. So, let us look at that suppose the sum is not 1. So, suppose you have maybe I will just put it in quotes here, maybe you have a three fourth here. And you have a one third here. So, we know that three fourth is greater than two thirds. So, if the probability that M accepts for a string that is not in the language is less than one third it is also less than one fourth.

Therefore, we can just pick that constant to be the minimum of whatever these 2 constants are. I mean, not the minimum 1 minus this number and this number. The second observation which is again, clear from the definition is BPP contains the class P, because what is a probabilistic turing machine, it is a turing machine which has 2 transition functions. But if you look at a deterministic turing machine, it can again be treated as a probabilistic turing machine, which were both delta 0 and delta 1 are the same transition functions.

It uses the pointers and it uses the same transition function, irrespective of what that value is. So, and therefore, if a language I mean it always basically outputs an answer with exact certainty, there is no probability involved. So, it is a major open question in complexity whether these 2 classes are the same or not. So, now, this is kind of counter intuitive to what we have seen so far.

So far, when we looked at complexity classes with varying computational resources, especially for I mean in the polynomial time setting, we saw that it is believed that those classes are not the same, but due to certain evidences that we have in the case of the class

BPP. Here again researchers believe that it is possible to de-randomize any probabilistic polynomial time computation that has some bounded error.

So, it is believed that these 2 classes are the same, but so, this is also another open question. So, is BPP contained in NP? So, this is also open. So, again it is believed that BPP is contained in NP, but that is not known. So, if NP is contained in BPP, then well, no, it would not imply but so, we will get actually certain results. So, we will see that, so we will see that in maybe in a couple of lectures that why this is believed to be true and not the other way around.

But as our frontier of knowledge stands, we do not know how these classes compare. So, therefore, even if he can show that P is equal to NP, it does not quite show whether BPP is contained in NP. See, so let me just talk a little bit more on this. So, the reason so why is it that we do not know this answer or why is it that even showing P is equal to NP, would not throw any light on this question is because if you look at an NP machine, so an NP machine in some sense can make an error.

In the sense that if you have a input which does belong to the language, you can have a computation path which rejects, but on the other side, if you have an input, which does not belong to the language, there the machine never makes an error**.** So, I will just put error in quotes, because it is not proper to use that term, but there the machine will always reject that input. So, that is the fundamental difference between what is happening in the case of a BPP machine and what is happening in the case of a non deterministic machine.

So, a BPP or any BP TIME machine, so, this has what is called a 2 sided error. Whereas, we can also consider probabilistic turing machines where there exists only one sided error. So, for strings that are let us say, not in the language, it will never give a erroneous answer and only for strings that are in the language it can give a erroneous answer with some certain probability. So**,** any questions?

So, what is happening in the case of BPP or in general, in the case of BP TIME, if you have an input which belongs to the language, you still can get an error with probability less than one third. In basically if you look at this statement, for x belonging to the language, the

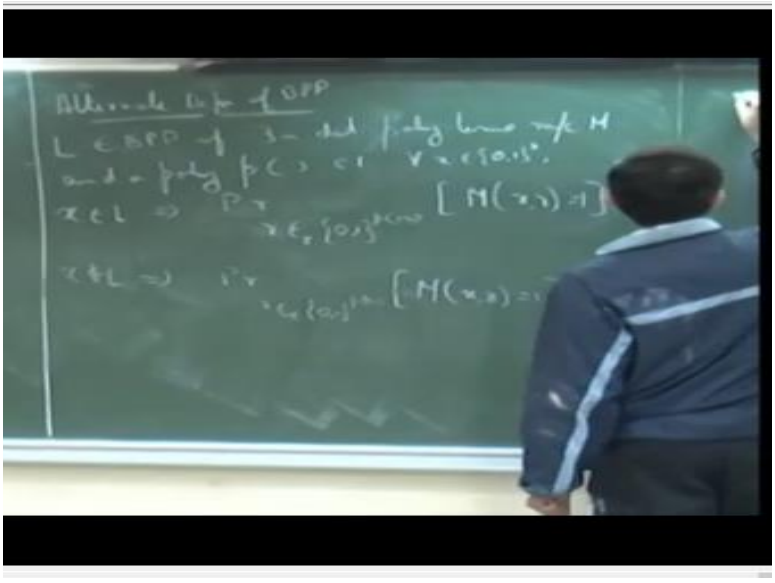
probability that M of x will be equal to 0 is not 0 that probability is not 0. It is only that that probability is less than one third.

Similarly, if for an x that does not belong to the language, M on x so what is the probability that m on that input x outputs 1. So, that probability is also not 0, what the only thing that we are guaranteed is that probability is less than one third. So, there is a nonzero probability that the machine can make an error for both types of inputs, whether that input belongs to the language or that input does not belong to the language.

So, there is a nonzero probability, the only thing that we are guaranteed is that that probability is quite small, the machine is not related to a sequence of coin tosses, but when you have an input that is fed to the machine and you look at the output of that machine on that input, that is with respect to a sequence of coin tosses and let us say you have a machine which runs for T steps.

So, then you can have 2 to the power T possible outputs, for all 2 to the power T, different sequence of coin tosses and now, this probability is basically computed over all such choices.

(Refer Slide Time: 31:08)



So, there is also an alternative way to define BPP basically, along the lines that I just said. So, we say that L belongs to BPP, if there exists a deterministic polynomial time machine M and a some polynomial P such that for all x if x is in L then the probability that for a for a random string r that is chosen from the set of strings 0 1 to the power P of x. And M given x, r outputs

1 is greater than or equal to 2 third and if x is not in L then the probability that again for r that is chosen uniformly at random is less than one third.

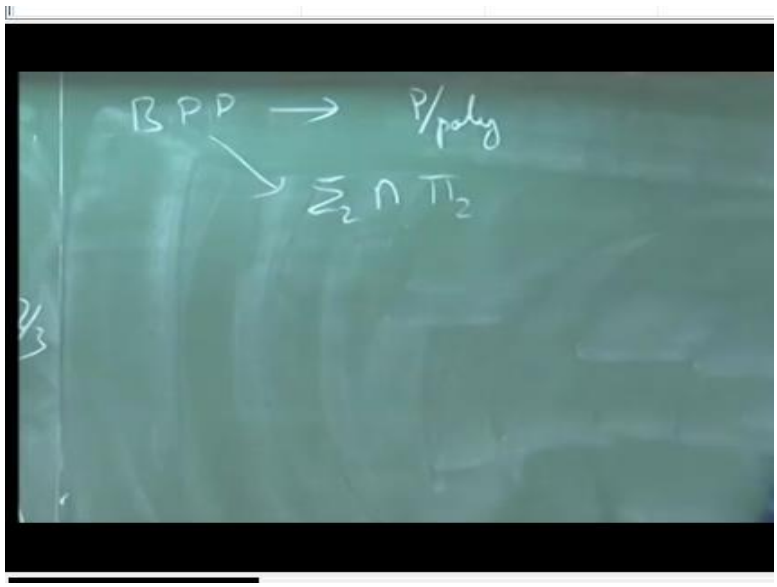
So, we can define BPP in this way also that there is a polynomial time deterministic turing machine M and some polynomial P. So, you can think of this polynomial as the polynomial corresponding to the running time of this machine. So, now, if you take a string x in L and you pick a random string of length P of x any random string. So, then the probability that M given x and r will accept is greater than two third. And if x does not belong to L, then M given x and r will accept is less than one third.

So, here the probability is coming over the choice of your random string. And as I said if a machine runs for T steps, then how many random strings can you have? So, it can be 2 to the power T. So, in this case it is 2 to the power P of x. So, this is also another way of looking at this class BPP. And you can actually prove that these 2 other same, exactly the smaller will correspond to the coin tosses at each step. And this notation just means that it is picked, it is sampled uniformly at random from this set of strings.

Of course, see, so let me ask, so maybe I should have this here also. In fact, you can. So, this is not known, but you can actually simulate BPP in of course, you can do it in x p, but you can also do it in P space. Because what you do is, you just cycle through all the strings of length P of x. And that will take only P of x number of tape cells. And for each such string r you simulate the machine and you store its output or you keep 2 counters, whether you and check whether that counter I mean, whether that machine outputs 1 or not. If it outputs 1, you increase one counter, if it outputs 0, you increase increment the other counter.

And then finally, you just check that for how many strings you get 0 and for how many you get a 1. So, that will allow you to compute the exact probability and then based on that you can accept or reject. So, there exists not only turing machine which can simulate a BPP computation, but a turing machine in this class P space. And as we shall see, not only is P space, an upper bound for BPP.

(Refer Slide Time: 36:35)



So, this we will see this after mid sem that BPP is also known to be contained in the class P by poly as well as in the second level of the polynomial hierarchy. In fact, in the intersection of these 2 classes. So, it is not known whether it is in NP, but it is known that it is contained in the second level. So we will see these results afterwards. So, coming back to this question of one sided versus 2 sided error.

So, now you can see why we call this 2 sided error, because in both these cases, there is a finite probability that we can make an error. But if you look at let us say, an NP machine, for strings, which do not belong to the language, the machine basically does not make any error along any computation. So, I do not think it is known to odd because, see, the problem that will happen is so this is the definition.

So, now suppose if you want to replace this definition with another statement, where you accept if it is, let us say, greater than 2 by 9 and you reject if it is less than 1 by 9. So, there is some gap but both these constants are on the same side of half. So, then the problem is that the moment you try to amplify these 2 constants, so I have to somehow show that that definition is equivalent to this definition.

So, the moment I tried to amplify that constant, that amplification basically occurs by running this machine several times. So, we see that day after tomorrow, how that can be done. Both these constants actually will simultaneously start decreasing based on what their value is. So, only if they had been on the same or if they had been on different sides of half. Would they have polarize to 0 and 1.

Since they are on the same side of half, as you said, they would get polarized to only one side. So, this still does not say that you cannot do it. But the usual way of showing that any constant I mean, two third and one third can be replaced by any constant does not work for any constant that is on the same side of half. So, that argument does not work.

Yes, yes, you cannot use that. So, we will see that day after tomorrow. So, maybe that will give you a better answer about why that would not work, when we see the proof of that. Because if you try to simulate that construction with constant that you mentioned, they would only get shifted to one of the sites. But**,** so maybe we should just hold on to that question. So, I will stop here today, stopping a little early but that is okay.

Now, but that is not allowed by the definition then you cannot say that that language is NP time. So, I say that a language belongs to this class, if there is a machine, which has this property, that language will be of course, decidable by that machine, but I will not be able to call that language to be contained in this class. So, only if the machine has this property, so this is a definition, and this is a class with some promise.

That under the promise that the probability is either greater than two third or less than one third only then can I say that the language is in this class. So, even if you decrease this constant, so, of course, the same machine would not work, but you can show that there is a different machine, which will accept the same set of strings and it will reject the same set of strings. So, that is what I was saying.

Yes, so that is what we will be able to show that one by 2 plus some positive constant works. One over polynomial it cannot be anything because if it is one over exponential that constant then it is the same as basically TP. But here we want by definition something that is not arbitrarily small, there should be some not trivial gap.