**Formal Languages and Automata Theory**
**Prof. Dr. K.V. Krishna**
**Department of Mathematics**
**Indian Institute of Technology, Guwahati**

**Module - 14**
**Introduction to Complexity Theory**
**Lecture - ~~0~~1**

# Time Bounded Turing Machines

**Formatted:** Space Before: 0 pt, After: 0 p
Line spacing: single

**Formatted:** Space Before: 0 pt, After: 0 p
Line spacing: single

**Formatted:** Space Before: 0 pt, After: 0 p
Line spacing: single

**Formatted:** Centered

**Formatted:** Space Before: 0 pt, After: 0 p
Line spacing: single

**Formatted:** Centered

So fa~~o~~r in this course, you have already got exposé to~~,~~ the notion, formal notion of algorithm and then~~,~~ the respective problems so called decidable problems. And in contrast~~,~~ we have also from theoretical point of you, you have already understood, how to observe a problem is undecidable, and you have understood, some of the undecidable problems. So, in the real word problems, the problems which are, you know~~,~~ which can be handled through algorithms~~,~~ that we are interested in, and the notion of algorithm~~–~~ as a turing machine; as the through a turing machine model, that we have realized and observed some of the problems~~,~~ which are of importance.

Now, in your algorithms course, you would have~~,~~ you know~~,~~ concentrated on the topics like~~,~~ you know the main concepts like, designing algorithm and understanding its complexity, complexity of an algorithm. It is a complexity~~,~~ you will be, you would have looked into various accepts particularly, if you look for time complexity, space complex city and within the time complexity, various classes, etcetera. Now, here what do we do?~~,~~ W~~w~~e have already understood for a to some extent that~~,~~ theoretically~~,~~ for what kind of problems you will be designing algorithms and some of the examples that we have identified.

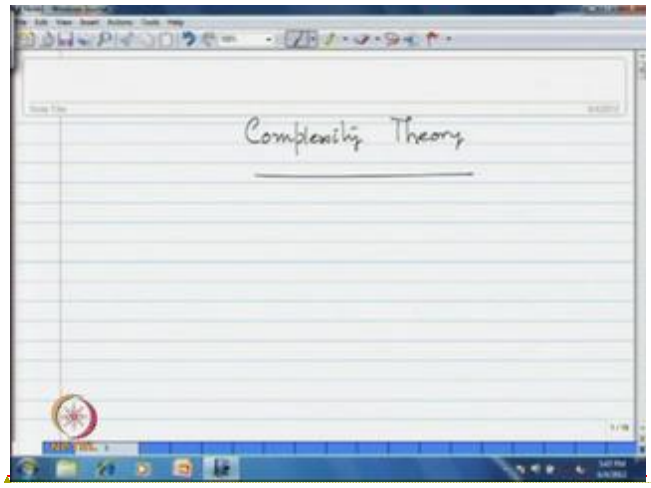
And~~,~~ we have also understood through this course that~~,~~ what are the problems that you cannot have an algorithm, so called undecidable problems that we have discussed.

Now, towards end of these course, now~~,~~ what we will look for is, you know~~,~~ some glimpses of complexity theory; that means~~,~~ whatever the algorithms that you have designed, may be~~,~~ here as a through turing machine model, some deciders that you have design.

Now, we will understand, what is the complexity of that particular design that you have done? In particular, we will look for time complexity; that means, in case, in case of algorithms~~,~~ the way that you have done in your algorithms course, you will be counting the number of computational steps and you are reporting with respect to the input parameter, If say for example~~,~~ n is the input parameter.

So, depending on what are the data set that you are choosing and some parameter n and with respect to that, you are reporting the complexity may be some t n s complexity of that particular algorithm. Now, here will go in a systematic way, I will introduce, how we actually report the complexity of a particular algorithm. Here, turing machine, and what is the notion and that related some glimpses of complexity theory that~~,~~ we will discuss in this course.

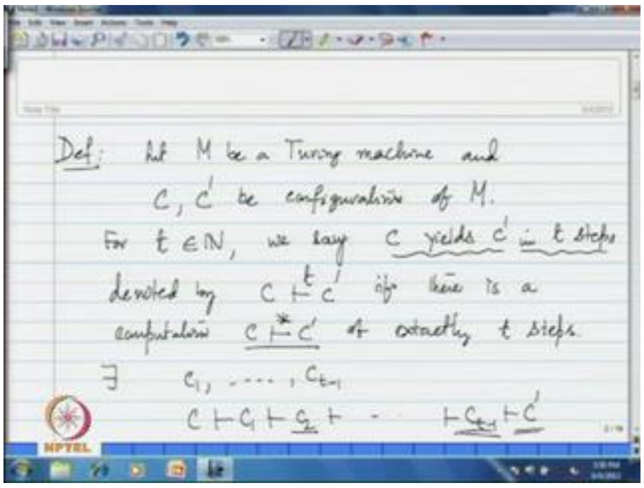
~~(Refer Slide Time: 03:07)~~



Complexity Theory

~~Here, first I introduce the notion of like, how we count the number of steps in turing machine.~~

(Refer Slide Time: 03:16)



Def: Let $M$ be a Turing machine and $C, C'$ be configurations of $M$.
For $t \in \mathbb{N}$, we say $C$ yields $C'$ in $t$ steps denoted by $C \vdash^{t} C'$ if there is a computation $C \vdash^{*} C'$ of exactly $t$ steps.
$\exists\ C_1, \ldots, C_{t-1}$
$C \vdash C_1 \vdash C_2 \vdash \cdots \vdash C_{t-1} \vdash C'$

Here, first I introduce the notion of like, how we count the number of steps in turing machine.

**Formatted:** Left

**Formatted:** Font: (Default) Times New Roman, 12 pt

**Formatted:** Left, Keep with next

**Formatted:** Left

**Formatted:** Centered

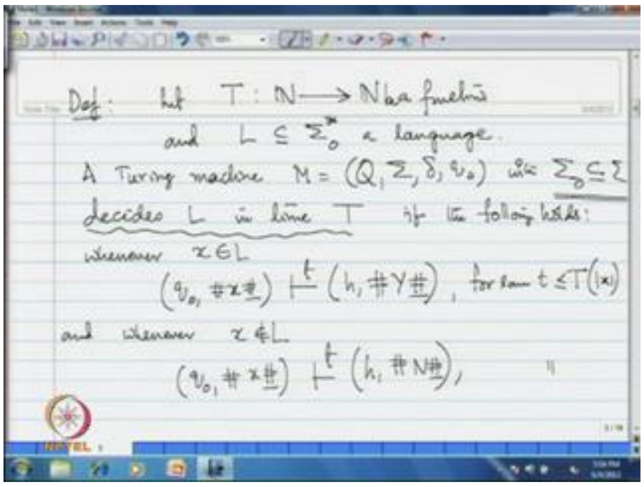
So, let me give this definition formally.~~ –~~ Let M be a turing machine and configurations be configurations of M. Now, for number natural number t, we say C yields C dash in t steps. So, from this term itself, you know, you can guess, what would be the definition, but let me present formally here, we say C yields C dash in t steps, that is what, I am going to denoted by, this notation is somewhat close to the once, which we have already discussed.

So, this… In t number of steps you have to get C dash from C. So, the naturally, the definition is, if there is a computation you know, the competition we right like this, there is a competition, the star of exactly t steps . So, when we have discussed computations you know, this C gives C dash infinitely many steps, that is what is known as computation and this computation that relation, that with respect to star, reflexive transitive closure of one step relation, we have discussed.

Now, this C should gives C dash, in exactly t number of steps, in which case and that relation symbol, I will write, what is the number of steps there. So now, we formally look at that this particular thing. So, between two configurations over various natural numbers, so depending on the number of steps, that it is taking to yield one to another, that we put on that particular relational symbol.

So, that is what is happing here, you have certain configurations in between say C 1,C 2,C 3 and so on… C t minus 1 such that, so that means, there exists some configurations such that, C gives C 1 in one step, then C 2 and so on… C t minus 1 and then you get C dash from that. So, you have exactly t number of steps, because this is the first step and there is the second step here, and in t minus 1 step you have reached here, and one more step to get C dash. So, exactly in t steps, if you can get then we say, we denote it like this, and C yields C dash in t steps. So, we are counting the number of steps that a turing machine takes to yield one configuration from another.

(Refer Slide Time: 06:34)



Def: Let $T : \mathbb{N} \longrightarrow \mathbb{N}$ be a function

and $L \subseteq \Sigma_0^*$ a language.

A Turing machine $M = (Q, \Sigma, \delta, q_0)$ with $\Sigma_0 \subseteq \Sigma$

decides L in time T if the following holds:

whenever $x \in L$

$$(q_0, \#x\underline{\#}) \vdash^{t} (h, \#Y\underline{\#}), \text{ for some } t \leq T(|x|)$$

and whenever $x \notin L$

$$(q_0, \#x\underline{\#}) \vdash^{t} (h, \#N\underline{\#}),$$

Formatted: Centered

So, this notion we will use, now to introduce the following, let T from natural numbers to natural numbers, a function, be a function, take a function and a language or some alphabet which is not containing blank, so we are writing sigma naught,~~–~~ _a language . So, take these things. Now, a turing machine say M equal to ~~(~~Q, sigma, delta, q naught~~)~~ with the sigma naught is containing sigma, a turing machine decides L in time T. So, what we are defining, when do we say, a turing machine M decides L in time T.

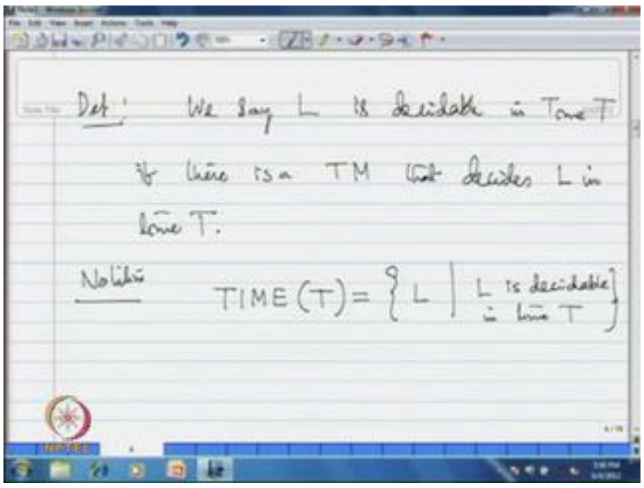
So, if the following holds, what is that, whenever x is in L in that language, then if you start the machine with x as input, so that is the initial configuration; in t number of steps, if it should halt by saying yes, because this is in the language. So, this t, for some t, this number is less than equal to T. So, the input parameter is essentially we have taken the length of the input string and whenever, this string is not in the language, the turing machine has to say no within the time bound steps. So, it will say no, if it is not in the language.

So, this is the output, but the number of steps it is taking for some t less than equal to T mod x. So, that means, you look at here, you are given a function t and a language, we say a turing machine M, of course, the language you should be able to fit the strings of L, should be able to fit to the turing machine. Therefore, we are taking the condition that, sigma naught is containing sigma, this particular condition.

And now, we say this turing machines decides L in time T, if the following condition holds, what is that condition, if x is in a language, the decider has to say yes, you know that, and if x is not in the language, you know, the decider has to say no, for the particular that string. Now, it is not only that, we are putting a time restriction on it, with respect to the function t given to us, what type of time restriction.

So, what are the input that you are giving x, the length of x is the input parameter, if you call that is number say n, n length input if you are giving, the number of steps, it is taking to say yes or no whatever, depending on whether it is in a language or not, you should take within t of n steps. So, this is the condition that we are putting and accordingly, we say that, this t within this time, we say L is decidable in time in this.

(Refer Slide Time: 10:39)



Def: We say L is decidable in Time T if there is a TM that decides L in time T.

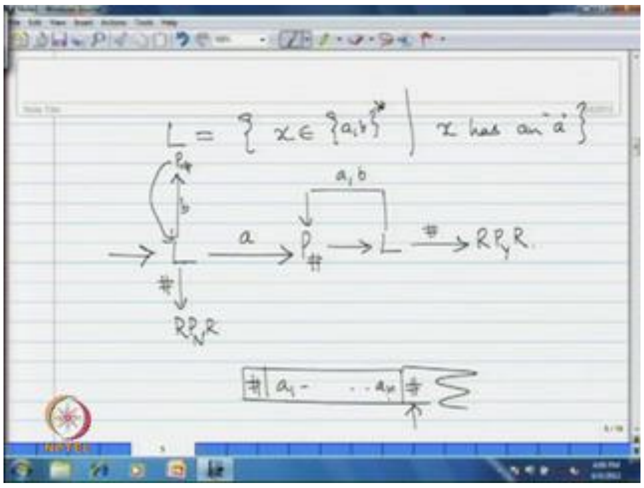
Notation $\text{TIME}(T) = \{ L \mid L \text{ is decidable in time } T \}$

Formatted: Centered

So, we will give that definition also formally. We say L is decidable in time T, T is the function from natural numbers to natural numbers, if there is a turing machine, this is the turing machine that decides L in time T. We have already define, when do we say turing machine decides a language in time T, where T is a function from natural numbers to natural numbers that we have already mentioned. Now, if you take an arbitrary language and time T we say, the L is decidable in time T, if there is a turing machine, the way that we have mentioned, if there is a turing machine which decides L in time T.

So, this is now, I let me give on notation here and the notation is, I use this time T, if you are given a time function T, the time T I mean, those languages which are decidable in time T. So, time T we mean… So, you collect all those languages which are decidable in time t; that means, for each has for each such language you should have a turing machine and we should decide that particular language in time T the given time function T. So, time T is that.

(Refer Slide Time: 12:28)



$L = \{ x \in \{a,b\}^* \mid x \text{ has an } a \}$

$$\begin{tikzcd}
 & R_{\#} \arrow[d, bend left] & & & \\
{} \arrow[r] & L \arrow[u, "b"] \arrow[r, "a"] \arrow[d, "\#"] & P_{\#} \arrow[r] & L \arrow[r, "\#"] \arrow[l, bend right, "{a,b}"'] & RP_YR \\
 & RP_NR & & &
\end{tikzcd}$$

$\# | a_1 - \cdots a_n | \# \ldots$

Formatted: Centered

Now, let me give some examples. A very simple example let me consider, say if you take L to be set of all those strings say over (a, b) for convenience let me consider like this, such that x has an a, the symbol a, you know this is the regular language, this is very easy to construct a decider for these. Let me consider this particular decider–_say, what do I do, first I will take a left move, I hope you are following this input notation, that we have already mentioned in the definition.

So, you take an input say, a 1 a 2 a n on the tape and this is how we are starting and we keep cross checking. So, how I will design here, you take a left move and cross check whether this particular symbol is a, if it is a, you are happy, you continue further and you have to say yes, if it is not a, then you can cross check keep going to left and that is how you can would have design a turing machine, a decider to cross check that.

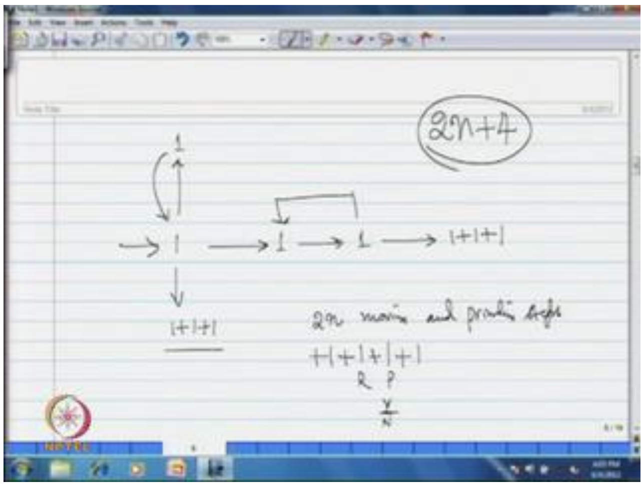
In the beginning, say for example, if it is empty string, which is not having a, so that means, is the beginning itself, you will have blank. So, in which case what do you do, you simply take a right move to and print no, because here, there is no a. Suppose in the beginning, if you have some b (s), you can erase that simply, if you have b, and then take a left move. So, keep doing these, if you are having b (s), only b (s) then what will happen through this particular branch, if all this are b (s), it will simply erase this, erase this and so on.

Till a, till the a 1, that a 1 is also b then you encounter blank, then you take a right move and print no and take a right move and halt, in the required format that would have decider like this, if you have a at some point in between, then we have to print yes for this. So, any way, this also we will erase, then take a left move and there after whatever that you get, whether a or b because you have got at least one a. And therefore, you will keep continue to erase the remaining symbols and when you get blank finally, you take a right move, here you print yes and take a right move. So, this is a very quick decider that you would have design and it is easy to consider.

Now, let us look at this decider, how much time it is taking to decide this language. Now, we have mentioned that each transition will be counted as one step; each transition will be counted as one step. Now, let as look at the transitions here, although we have given a turing machine, which is composition of some primitive machines. Now here, each machine, how many primitive steps that it has, we can easily understand and count the number of steps, number of transitions.

Look at in the first step; I am taking one left move; that means, here your taking one move; that means one transition here in the beginning. If it is blank, so you are cross checking, then we are taking a right move printing n and going taking a right move; that means, in this branch essentially, it is taking here one step,~~;~~ here one step.

(Refer Slide Time: 16:11)

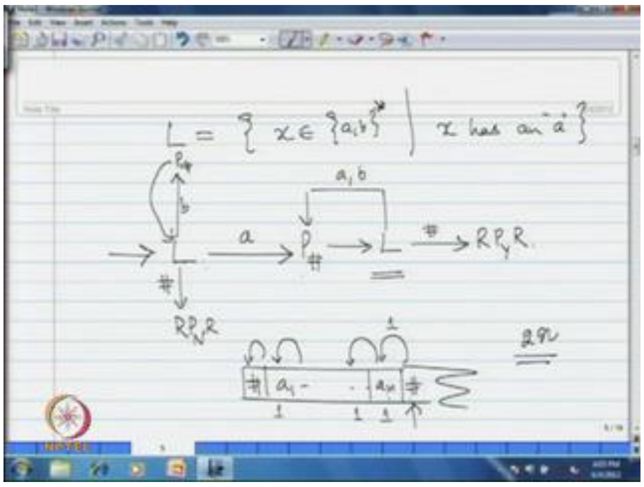


Formatted: Centered

So, let me write. So, here this is one step and when I take this branch, it is taking 1 plus 1 plus 1, three steps essentially it is taking. This for right move on step, printing n that is one transition, that the turing machine and taking right step, so there are three. Now here, if it is, if you are going to this branch, here, you are having one step, and you are connecting back to this. So, here you have a loop and every step here, you are printing a blank symbol, fine.

Now, when you go to go through this branch, here again, you have one step, printing step and here moving step. So, here printing step and here, moving step. So, that is here one step here, and one step here, I am just putting the count and you have a loop here, and then when you proceed to this branch and again, here you have 1 plus 1 plus 1 steps. So, these are the primitive steps in a turing machine that you have.

(Refer Slide Time: 17:17)



Now, how to report these numbers? Because wherever you have the loop, now you look at, in this particular. Suppose, if you look at the tape, what you are doing, from here you are coming to this particular position in this cell. So, you are taking one left move and then you are erasing it, whatever it is, whether it is a or b. So, in this particular cell, whether it is a or b, whatever it is, if you look at the scheme, what that way, that we have design. In this particular cell, you are simply erasing, so that means, one left moves in step and one erasing step that you will have.

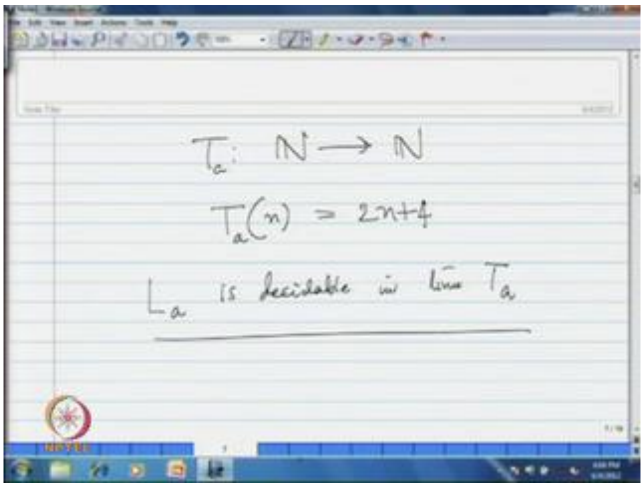
So, here let me write one step you have here, and then we are taking one left move and whatever it is, you are erasing here and so on; so that means, what is the happing here, you are taking a left move and printing something, that is erasing essentially. So, taking a left move erasing; that means, for each of this, you know till this, you have two n steps that you can quickly see, by the time you reach from here to here, you know you have to make n number of moves and you know n number of printing steps; that means, two n number of moving and printing steps. So, you can 2 n moving and printing steps.

So, in fact, n number of moving steps, n number of printing steps till this point n. And then, you take one more move, left move here, you have uncounted blank, that is what is you are reading. So, at this position you are taking only one. So, plus 1 and then you take a right move and appropriately you will print either yes or no, depending on whether you have a or not. So, that printing step is 1, printing. So, you take a right move, take a right

Formatted: Centered

move and then print 1. So, this is for left move, this is for right move, this is for printing either yes or no, whatever that you print and then take a right move. So, how many steps total that you have, that is 2 n plus 4 steps.

(Refer Slide Time: 19:25)



$T_a : \mathbb{N} \longrightarrow \mathbb{N}$

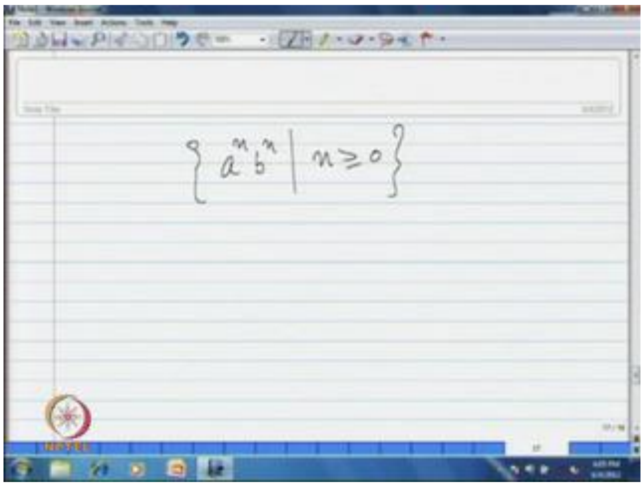
$T_a(n) = 2n+4$

$L_a$ is decidable in time $T_a$

Formatted: Centered

Now, if you take a time function T from natural numbers to natural numbers, given by T of n is equal (2 n plus 1)– (2 n plus 4). Then you see this time function, within this time function you know, the language L whatever is given here. So, let me call it as L a for the timing. So, L a is decidable, L a is decidable in time T a as mentioned here, because we have identified a turing machine, which is the decider, which is deciding this language and the time it is taking is 2 n plus 4 steps on any input string. If the input is having a, it will halt by printing yes, if the input is not having a, then it will halt by printing no, but the total number of steps it takes is 2 n plus 4. So, this particular language is decidable in 2 n plus 4 steps.

(Refer Slide Time: 20:39)

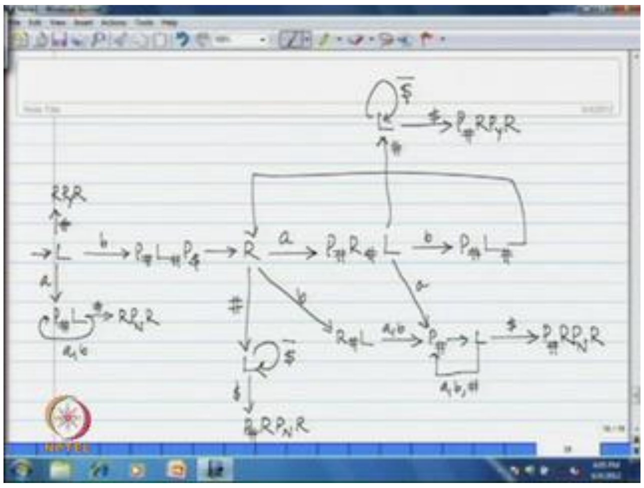


$\{a^n b^n \mid n \geq 0\}$

Formatted: Centered

Now, let me consider, lets example you know, we are very familiar with this, a power n, b power n such that, n greater than or equal to zero. If you take this language you know, this is a ((Refer time 20:59)) language, you have design a decider for this, I hope. And now, let me consider a decider for these and see how much time that the decider is taking.

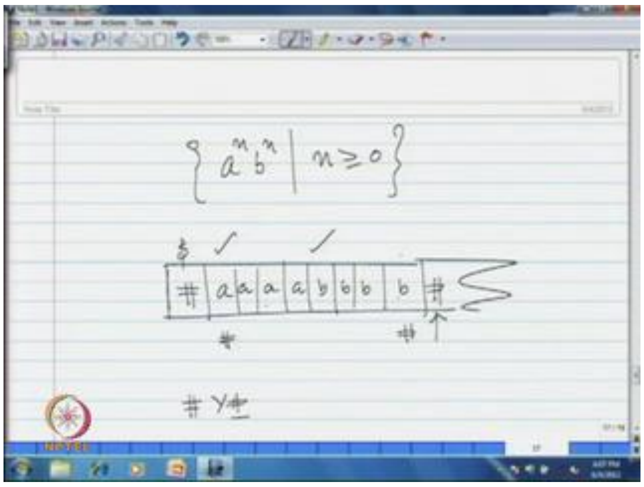
(Refer Slide Time: 21:13)


$\bar{\$}$
$L \xrightarrow{\$} P_{\#}RP_yR$
$\#$
$RP_yR$
$\#$
$\rightarrow L \xrightarrow{b} P_{\#}L_{\#}P_{\$} \rightarrow R \xrightarrow{a} P_{\#}R_{\#}L \xrightarrow{b} P_{\#}L_{\#}$
$a$
$P_{\#}L \xrightarrow{\#} RP_NR$
$a, b$
$\#$
$b$
$a$
$\bar{\$}$
$R_{\#}L \xrightarrow{a,b} P_{\#} \rightarrow L \xrightarrow{\$} P_{\#}RP_NR$
$a, b, \#$
$\$$
$P_{\#}RP_NR$

Formatted: Centered

Look, I have constructed a decider, if you sit, you can also constructed, we would have constructed by now.

(Refer Slide Time: 21:30)



Formatted: Centered

What is the logic I am using here is, if you take a n, b n let me assume, say for example, a string which is in the required form, when in the language, so this is where a starting, the logic what I am using is, I will take a left move, if I am encountering b, I will make it blank and go till this end and I will make it, I will come to write and match whether there is a, I will make it blank. But, one thing I have to remember, that once I erase the tape when I have to halt, I have to halt with this kind of information either yes or no, I have to halt like this, where yes is to be printed in the second cell, in this cell, but once, if I erase, if I am in between then I will not know, how many number of blank cells are here.

So, I have a possibility of a hanging. So, this kind of, you know, techniques and all those things, that we have already discuss earlier. So, in the first place, what I will do, when I reach to this particular left n, I will prints a special symbol. So, say for example dollar, I will print here, and then when I am, when I have to halt, that the end of the story,–_if I am in between somewhere middle, I will go till end; that means, till–_I encounter dollar, then appropriately I can handle. So, this technic I will use, so I will design accordingly, so what I will do.

So, some trivial cases that you can check, say for example, in the beginning itself, suppose if you are encountering this; that means, if it is empty string is given as input. So, you know the blank cell and in the second blank cell you are given this, tape is given like this. So, that particular thing you can handle. So, if you have blank in the beginning, if you take a left move, take a right move, print Y, R, because n equal to zero, a power n, b power n form. In the beginning, if you encounter a, I am taking this branch; I am erasing the entire tape and then printing no.

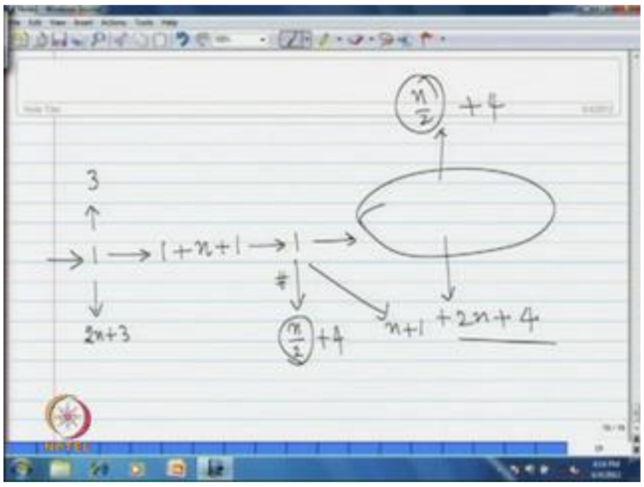
So, that is, what is this particular branch? And, if I am starting with b, then what are the mechanism, I have just mentioned, I will follow that; that means, I will print blank, take L hash, print dollar in the beginning. Then, I continue in the loop by cross checking corresponding to each b, whether there is a or not. So, if there is a, take a right move, print blank, take a R hash, take a left move if there is b, then print blank take L hash and continue in these loop. If the input is in the given format, a power n, b power n, then after cross checking a, for the b, then you will come to this branch.

Here, you will just go through the blank, so the blank. So, what I am writing here dollar bar, of course I know, they are all blanks and then finally, I will print yes through this branch here. Otherwise, I gave the exits laws here. For example, at this branch if you are not encountering a, you may encounter blank, you may encounter a blank symbol, when R b. If you are encountering blanks symbol; that means, corresponding to b, there is no a, there appropriately we are, you know printing no, if you are encountering b, the similar way that I have handle to print no.

And in this situation corresponding to, you know b, I have encounter a, and I went R hash, when I take a left move, if I encounter a, it is similar to, you know, you are having, it is not of the form, a power n, b power n. So, there also I am printing n. So, the decider is design like this, this kind of things we have discussed earlier as well. So, now if I consider this decider to decide this language, a power n, b power n.

How many steps it is taking, that is, what is the time function associated to this particular turing machine, which is deciding, a power n, b power n. So, let us look at that analysis. Look, in the very beginning here in this branch, it is taking at this one move and if you go to this upper branch, where you are encountering blank, there you can clearly see, there are three moves. So, as I had written earlier, if you write like this, then things will be little bit clear to you.

(Refer Slide Time: 25:49)



**Formatted:** Centered

So, here I, it is taking one step and here, you require three steps. If I go to this branch, what is happening here, you have to erase, if you encounter a, you have to erase everything and after that, you have to print no. It is then as explained earlier, you have to erase everything. So, after this one step you require the, 2 n plus 3 steps, if you take this branch, here you are printing blank, then L hash. Printing blank means one step, L hash means you have to come all the way to the left hand, so n number of cells, input length of n.

So, you have to take n number of steps and then you will take one step to print dollar there, then right move, right move takes one step. And, there after you continue in a loop and see, if you look at these branches, if it is the case of no, if it is not on the given input format, in this branch if you carefully observe you know, you are encountering this branch means you are almost like you know, you have done some matching through this loop. And at the end of that you know, for corresponding to a b, you do not have an a, but you have erased everything.

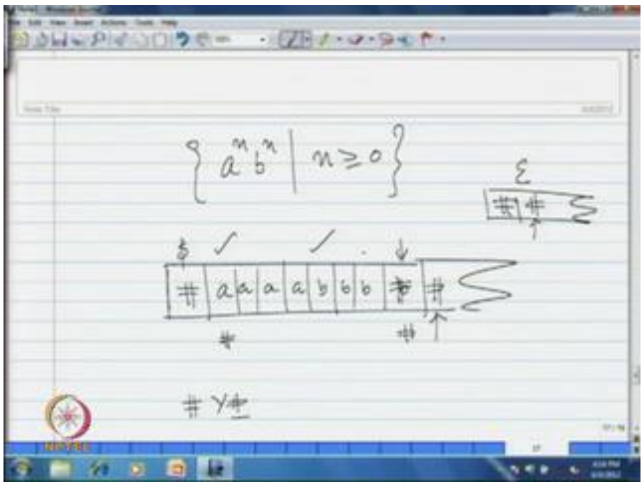
So, roughly you are in between the input; that means, roughly n by two cells, I had from the left hand. So, I may say roughly, for roughly to say, I will put a circle symbol here. So, roughly n by 2 steps that you take, and then what do you do, after taking n number of n by 2 number of left moves, then you take, you will dollar, you will print again back to blank, then take a right move, then print no, then take a right move. Here, there are four, four steps, so ~~(~~n by 2~~)~~ plus 4 steps in this branch.

If you go to this one, R hash, R hash means you know, from this position you have to go to the right end, but I am not very clear that, how many cells so far, I have erase or whatever. So, in the worst case what will happen, in the beginning itself, suppose if you have the miss match. So, the length of the input whatever is given, so that much you have to go, I am a say now here, when I say R hash. So, n number steps that you have to take plus 1, that is corresponding to taking a left move and thereafter you continue in this loop to erase, that is 2 n steps and then here again 4 steps.

So, now whatever they exit Y a a, that will connect here, the 2 n plus 4, these are the steps, that it will take. Now, when I am exiting through this, that means, the input is in the required format; that means, you are completely erasing the tape and you are going here. Of course, in each case we will be erasing the tape completely, but you see that from a, through this branch when you are exiting the 2 n plus 4 steps here. Similarly, when I am exiting here, you see that you may be roughly half of the tape, because you are stopping your going towards saying yes.

So, roughly you are in the half of the tape. So, here also this will continue. So, half the size and then plus 4 steps, because this dollar you have to converted it to blank, then taking a right move, printing yes, taking a right move. So, there are 4 steps here, n by 2 plus 4. Now, this loop how many times it is, it will run, you know that if it is continuing for the entire input to reach to the middle, which is the positive case here or you know, it will continue and taking to this blank side also, then ~~(~~n by 2 ~~)~~ plus 4 steps here, ~~(~~n by 2~~)~~ plus 4 steps . So, this is the, these are the extreme cases that you will be continuing and pursing the input.

(Refer Slide Time: 30:02)

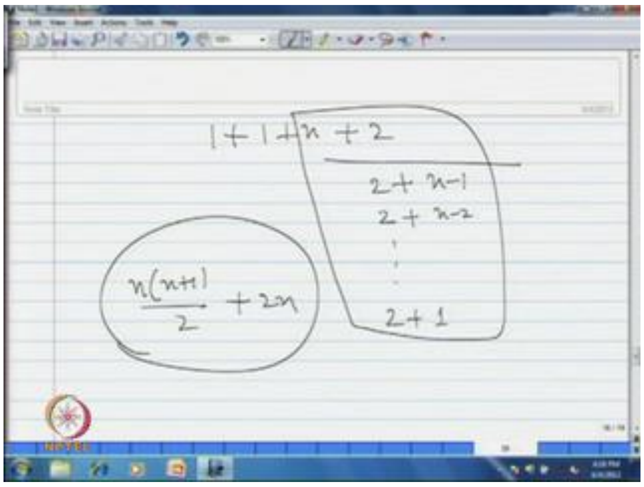


Now, let us see, like now many times this loop is running and how actually it is input getting decrease. So, if you carefully observe that, for example here, you make it blank, then you go till this left hand. So, n number of steps we have counted, then here you are printing dollar and taking a right move, two steps and then you are making blank here. Then, you are going till these points, till this point, here. So, this is blank at that time then you take a left move.

So, here n minus one step, your n minus one length you are going, when I am saying that here R hash, here n minus one steps that you have taking then you print blank there, then L hash, that time you have to take n minus two steps. So, in this loop, what will happen? So, here you have taken for L hash n steps, here n minus one steps, here n minus two steps and it will keep on decreasing, that you will come to the middle, come down to one. But, each of this you have a constant time added, that is you know, the printing and taking a move, printing and taking a move, the printing blank and taking a left move to cross check.

Formatted: Centered

(Refer Slide Time: 31:10)



$$1+1+n+2$$

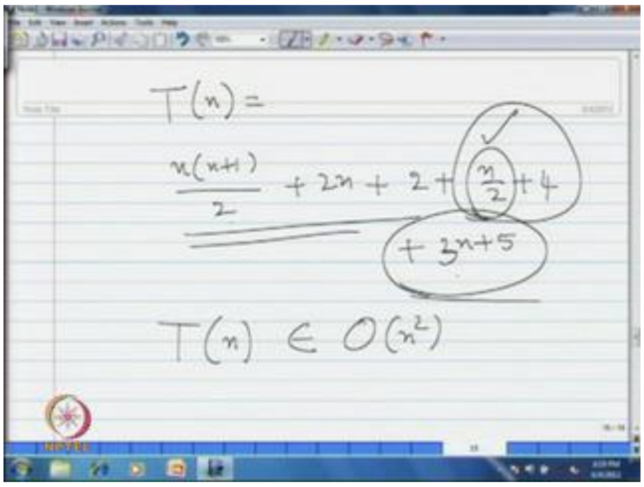
$$2+n-1$$
$$2+n-2$$
$$\vdots$$
$$2+1$$

$$\frac{n(n+1)}{2}+2n$$

Formatted: Centered

So, here you can observe that, the pattern is in the beginning; we have already noted this, 1 plus 1 plus n plus 2 there. Thereafter, in the loop you have count like this, 2 plus n minus 1 and 2 plus n minus 2 and so on 2 plus 1, so coming down till this. So, if you note this, this particular sum, you can quickly calculate, because here 1 plus 2 and so on plus n, that is, ~~(~~n into n plus 1 by 2~~)~~ and you have n number of twos that is 2 n. So, by including that, you can see if you exclude this, of course this much only in the loop. So, this is the amount of time that you are spending in the loop.

(Refer Slide Time: 31:59)



$$T(n) =$$

$$\frac{n(n+1)}{2} + 2n + 2 + \frac{n}{2} + 4$$

$$+ 3n+5$$

$$T(n) \in O(n^2)$$

**Formatted:** Centered

Now, if you carefully calculate all the time. So, we can report that n into, the total, the time, ~~(~~n into n plus one by 2~~)~~, plus the 2 n, 2, n by 2 plus 4. Of course, here if act in a positive sense, we are getting n by 2 steps here. So, this much time, if you are going this branch or this branch, this is where you will get the maximum time, because you are consuming the entire input in this and you are coming here. Otherwise, what will happen, in between suppose this happens.

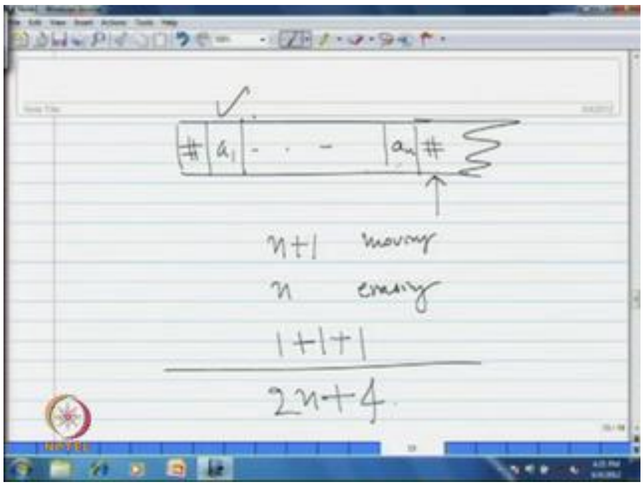
Then, in this branch you may not complete the loop, like you know going till middle, but you will be taking one of these branches, there we have already counted, the time it is taking you know, n plus 1, 2 n plus 4. So, the majority of this is actually adding here, there it will happen. So, are you know, after completing that much that many cycles, if you are doing this. So, this is in the worst case, this is also in the worst case after completion completing almost all the cycles, you are left with only n by 2, but whatever it is, it is order of n, here also it is if a order of n.

So, what you, what do you know say, that this 2 n plus 4, here you can see, the 3 n plus 5, this is what is the time it is taking. And in the loop we have a understood that, it is of order, n into n plus 1 by 2 plus 2 n. So, all this amount suppose if you had, that 2 n, the 3 n plus 5 or this n by, n by 2 plus 4 or either this component or this component will be added. So, the maximum is 3 n plus 5, so this component plus this component.

Now, suppose if I define a function, which is a quadratic. Now, I say this is, now of order, n square to loosely say or if you strictly define, T of n is equal to n into n plus 1 by 2 plus 2 n; that means, essentially 5 1 plus 7, something like this, if you define, what are the branch you take in this machine, either this branch or through this branch or through this branch, wherever that it is halting. This is the maximum time, is the maximum time it is taking, is the maximum time it is taking and this function you can quickly see, it is of quadratic time.

Thus, I can conclude that this language, a power n b power n you know, is decidable through this decider in quadratic time. In fact, I, we could come up with a precise time function for this, by counting appropriately. So, through this example, this is two examples. So, I hope you understand the concept that, when I am taking about the time bounded turing machines. So, the time bounded turing machine means, a turing machine which is you know, bounded with a time function; that means, a decider which is bounded with a time function. So, this time bounded turing machine concept that, we are talking to evaluate the time, it is taking to decide a particular language.

(Refer Slide Time: 35:31)

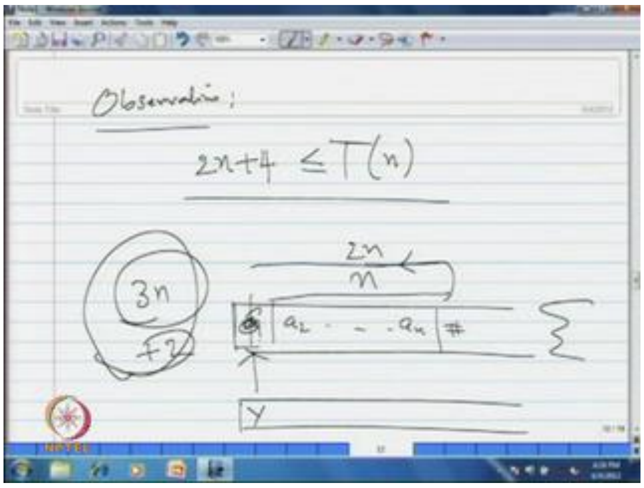


Now, let me come up with one more important point here, whatever is a time bounded turing machine that you consider, what is the minimum amount of time it takes on a input of length n. This is actually close to the first example that we have discuss, because if you are given an input of length n, whatever is the parameter that you want to cross check, whether it is, a power n, b power n, or you know having at least one a or you know having a b a as sub string or whatever.

Whatever is the situation, what we have to do, you have to erase the entry input; that means, you have to traverse from here, you have to take a left move, you have to erase it; you have to take a left move and erase it. This is only way that you have to traverse through this. So that means, you know, you have to take n moving steps to come to the left hand. In fact, to come to the left hand you know n plus 1 moving steps that you require.

And then, these n cells you have to erase; that means, n erasing steps and these are moving steps to come to this n, these are any where required. And after that, depending on the parameter that you are cross checking, you will take a right move and then here, for right move means 1, either you print yes or no in this cell; that means, one step and that take a right move and halt. So that means, what is happening, this is 2 n plus 4 steps or minimum to decide any language.

(Refer Slide Time: 37:13)



So, what I can conclude here, if you are talking about a time bounded turing machine, which is deciding a particular language, that any time function of a time bounded turing machine, this T (n) is always greater than or equal to 2 n plus 4. Because this time function has to take at least 2 n plus 4 steps to decide a language. So this, what a important observation. So, this is as per our convention, because we have fixed

Formatted: Centered

convention, that we will be sorting from the right hand on the tape and you will be you know, arising the tape and continuing the complex city.

Now, ((Refer Time 37:49)) suppose if one wants to consider the tape you know, say for example, there you do not require this kind of special symbol or anything. Say in a 1, a 2 and so on, a n, suppose this is given to you. And your convention, for example, if you want to start from the left hand, now what happens, you have to come back and print is not somewhere in between, you have to say yes or no and the tape say for example, in the first cell or whatever.

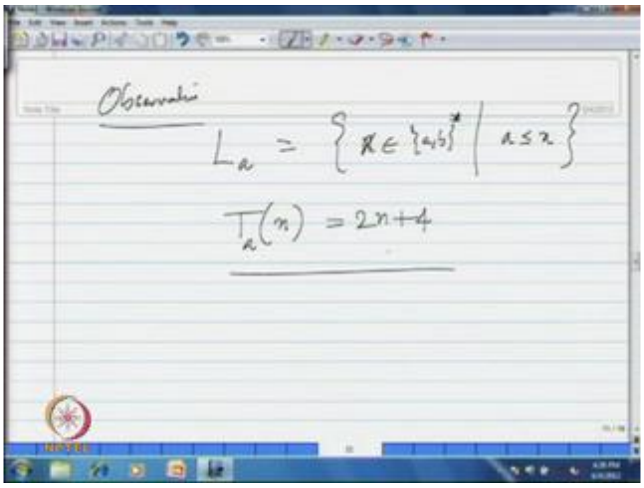
So, then in which case what you have to do, here you have to put some marker. Say for example, in the dollar then, you have to traverse all this way; that means, n number of steps it will take. Because, what are the parameter that you want to cross check then, when you are coming back say for example, if you are erasing. So, you erase all those symbols, till you come here. So, that means, this is 2 n number of steps. When you are combing back, you have to traverse all this n number of cells and on the way, you have to erase either, while going you are erasing or while coming you are erasing, whatever it is.

So, essentially to go here to this end and come back and erase the tape, you require 3 n numbers of steps. And after that, whatever that the parameter that you wanted to cross check, depending on the language; that means, you know, if I am starting from the right hand, I am taking minimum 2 n plus 4 steps to print yes or no. If I start from this side, I have to take 3 n steps to come back and again to print yes or no or whatever. Again, the required number of steps may be two or whatever. In the first cell, if you are printing yes and there itself, if we are halting or whatever.

So, you look at, you compare these two quantities. This is say for example, 3 n plus some constant and this is 2 n plus 4. Anyway, whatever is this constant, this 2 n plus 4 is any ways smaller than, this 3 n. Of course, a syntactically these two are equivalent, but you look at, the time that you are gaining at least you know, n number of steps that you are gaining. So, this is through this observation, I give the proper justification, when we have started discussing you know, the from the tape right end.

Because, when I have discuss the turing machines, I said like, you know, we will be starting, we will be giving the input format, that we will start from the right end. Now, if you start from the right end, you can quickly see that, you can gain n number of steps, when you are discussing the time complexity. So, this is one justification that I am starting from the right end and to talk about the decidability or accepting the string and etcetera in a turing machine.

(Refer Slide Time: 40:35)



Observation

$$L_a = \{ x \in \{a,b\}^* \mid a \leq x \}$$

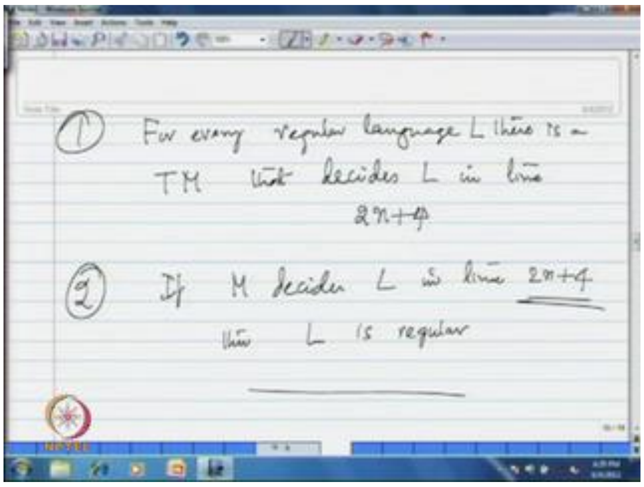
$$T_a(n) = 2n+4$$

**Formatted:** Centered

Now, one more important observation in this particular context. The first example, I have discussed L a; that means, all those strings which are having a as sub string. So, a b star, this is the language, today I have discussed, that a, is a substring of x. So, there is atleast one a, in this x. We have observed that, there is a turing machine which takes time. So, T a, I wrote to the time for this, is 2 n plus 4. And, just now I have mentioned that any time bounded turing machine, any turing machine which decides a particular language in a time ,t the time bound is you know, is the lower bound is 2 n plus 4.

So, 2 n plus 4 is less than or equal to, that particular time. Now, you see for this particular language, what are the decided that we have designed, that is of 2 n plus 4 times; that means, it is optimal, that is optimal time. So… And now the point is, this is the regular language that we have constructed a decider appropriately and we have decided that in time 2 n plus 4. This is the minimum time possible with any turing machine that we have.

(Refer Slide Time: 41:56)


① For every regular language L there is a TM that decides L in time $2n+4$

② If M decides L in time $\underline{2n+4}$ then L is regular

Formatted: Centered

Now, you can in fact, observe that, if you take any regular language. For every regular language, you can have a time bounded turing machine that, whose time is 2 n plus 4. There is a turing machine decider of course, that decides, for every language say L that decides L in time 2 n plus 4,— in time 2 n plus 4, this is the time function I am giving. Now, how do you make out this you know, if it is a regular language, you can manipulate via states only. You can, you have to just scan through the tape only once that you know.

And, when you are designing a turing machine, appropriately you design and whatever that you wanted to cross check, we just use via you know, the you can you can via states that you can maintain and you scan through the tape once, and may be for example, if we are starting from the right hand, you just scan through the tape. We know that, if L is regular, L power R is also regular; that means, if use reverse every string in the language that is L power R we are talking, so that is also regular.

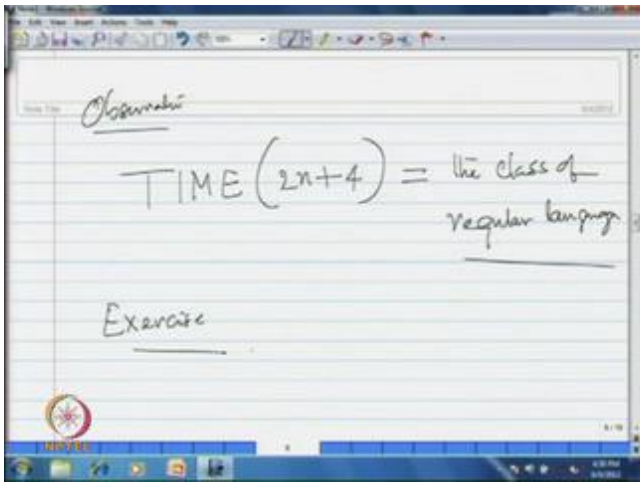
So, there … Since, regular languages are closed with respect to reversal either, you read from left side or right side one and other is same, now, the corresponding language that we are looking at. So, you just scan through the tape once and finally, we just take that three right moves, while you are printing yes or no. So, the point is using that particular mechanism; that means, what are the way that you design your d f a corresponding to this regular language, you can appropriately you know, design the states for a decider.

And in fact, you can design a turing machine, which decides this regular language in time2 n plus 4. Now, what about the converse, because 2 n plus 4 is the minimum time, I had mentioned now. If you decide a particular language in 2 n plus 4 times; that means, you just scan through scan through the tape only once and decide the things. Because, you do not have any other option, if I say, you are deciding a language in 2 n plus 4 time as you have observed. You are just scanning the tape only once and whatever that you wanted to remember, you have to remember through states only.

So, now the question is, if you look for the higher little— high higher order languages; that means, higher languages like you know, context free or context sensitive etcetera, what is happening, there you required some other auxiliary input or something else, but for regular language, you do not require. So, is it only, the language if for which you require 2 n plus 4 times; that means, if you have a turing machine which decides a particular language in 2 n plus 4 times, is that regular that is a question. In fact, it is so.

So, what I am saying is, if M divides M decides a language L in time 2 n plus 4, then L is regular ((Refer Time 45:10)). So, very important observation here; that means, L is regular, if and only if, there is a decider, that decides L in time 2 n plus 4 times. So, this 2 n plus 4 times, the minimum time is you know, for regular languages for the class of regular languages you have.

(Refer Slide Time: 45:35)



Observation

$$\mathrm{TIME}(2n+4) = \text{the class of regular languages}$$

Exercise

So, what I can now, write here, through this observation. Time as I have introduce the notation time 2 n plus 4, because here I am writing the time function is equal to the class of regular languages, the class of regular languages. So, this is the very important observation.~~–~~ Now, you can take it as an exercise to prove both the parts, because I have just discuss the ideas only, you take it as an exercise and prove both the parts; that means, you know L is regular, if and only if, there is a turing machine that I that decides in time 2 n plus 4~~,~~ an exercise.

Formatted: Centered