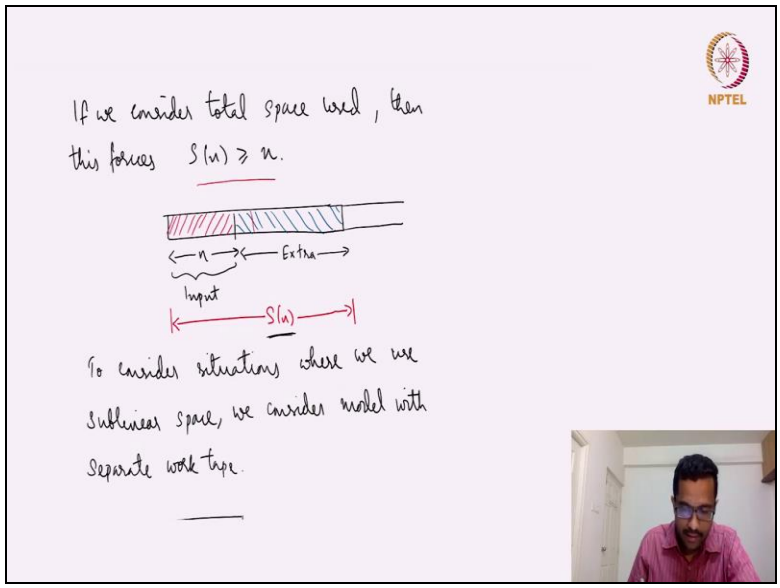
Computational Complexity
Prof. Subrahmanyam Kalyanasundaram
Department of Computer Science and Engineering
Indian Institute of Technology, Hyderabad

Lecture -14
Introduction to Space Complexity

(Refer Slide Time: 00:15)



Hello and welcome to lecture 14 of the course computational complexity in this lecture we will introduce space complexity. So, as I said at the beginning of the course the 2 most important resources that are considered in the in complexity theory or in the case of turing machines or in the case of computation in general is our time and space. So, we have been seeing P, NP, polynomial hierarchy and so on which are all complexity classes which are based on time.

Now we will see complexity classes where the resource under consideration is space. So, just like we saw time bound complexity classes we will see space bounded complexity classes. Some of the classes that we will see are L, NL and P space. So, L and NL are lock space and non-deterministic log space they use order log n space and P space is the space equivalent of the complexity class P and which is which is time.

So, P space uses polynomial space whereas the class P uses polynomial time. So, before getting into the actual nitty gritties of what L n, L P space etcetera are let us try to understand what does it mean when we say space usage of a turing machine. The case of time the usage time usage is

very clear because you just measure time in terms of number of steps that a turing machine takes to do a certain computation what is the space usage.

So, it there is a bit of certainty there. So, suppose a turing machine is given this is a turing machine type and let us say. So, as always n is the usually considered to be the length of the input. So, the input is there in n tapes or sorry n cells of the tape and as part of the computation maybe you have to use some extra space. So, the part in blue is maybe some extra space that is you being used for the computation um.

So, n is the red part is where the input originally was and some more space is being used in the tape. And and even the inputs where the input originally was even the red places could be over written because there is no requirement that the input remains as it is sometimes like sorting we just we do not retain the original order of the input. So, we may we may overwrite it that is because as long as the at the end we decide or compute whatever we want to decide or compute it is all fine.

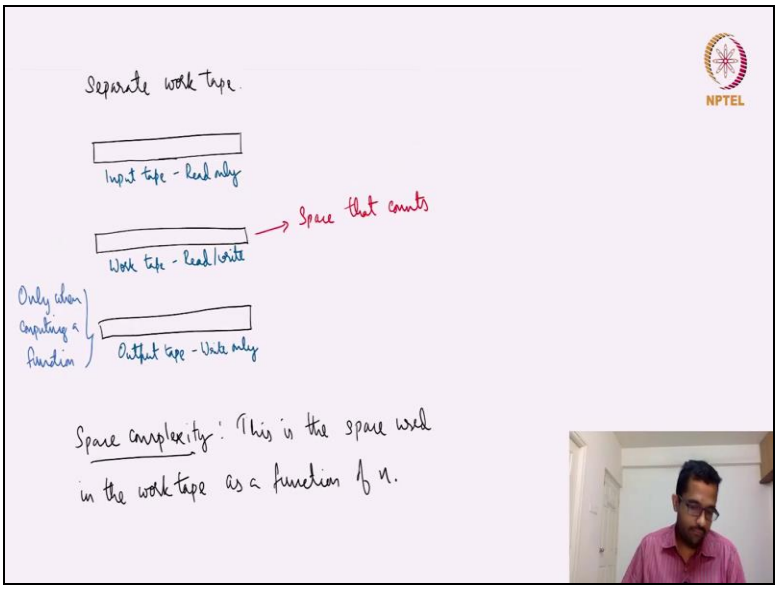
So, and let us try to do something naive let us say the space usage of a machine is a total number of total amount of cells that is used for the computation in the tape. So, suppose this is the tape. So, n is the number of; the amount of cells used for the input and some more extra and the total space usage of the steering machine is n plus extra which we will use S n to denote S n to denote.

So, this sounds reasonable. Now because if we use S n space which is n plus extra and we are counting accounting for essence we are accounting for exactly that. So, it seems ok but the problem is that in this model by the very definition of this model we have to always have S n to be at least n because the space usage always takes into account or the space usage is always at least takes into account the space required for to show the input.

And the space required to store the input is n bits or n symbols. So, the space usage ends up being at least n. So, now is it is it correct to kind of assign a cos of n because just because the turing machine itself the input itself uses n n symbols. So, what is the extra? Very less number of bits is used for the extra. So, maybe just maybe just login bits is extra space is required for the

computation but just because the input required n n symbols n plus log n is again n dominates log n.

(Refer Slide Time: 05:08)



So, we will say order in space but then the actual extra space required is only log n. So, here the problem is that S n in this model S n always is at least as big as n which is the input length. So, this model is undesirable for this reason it is ok when the space extra space uses more. But when the extra space uses less this is actually penalizing the machine needlessly because the extra is what we are interested in.

So, how do we account for login space or something that is less than n space or in general sublinear space? Because this space n is always available for the machine you can always use n space. So, we want to we want to have n symbols for the input but at the same time not be able to use that for extra computation. So we will have come up with a separate model. So, separate model is following you have three tapes or maybe 2 tapes sometimes third tape.

So, the first is a input tape second is the work tape and the third is the output tape. So, note that the output tape. So, there are 2 types of compute problems that we may have and most of what we have seen in this course so far are decision problems. If there is a decision problem we will not be using an output type we just have to accept or reject decision problem is like does this graph have a cake leak is this graph connected is this matrix full rank.

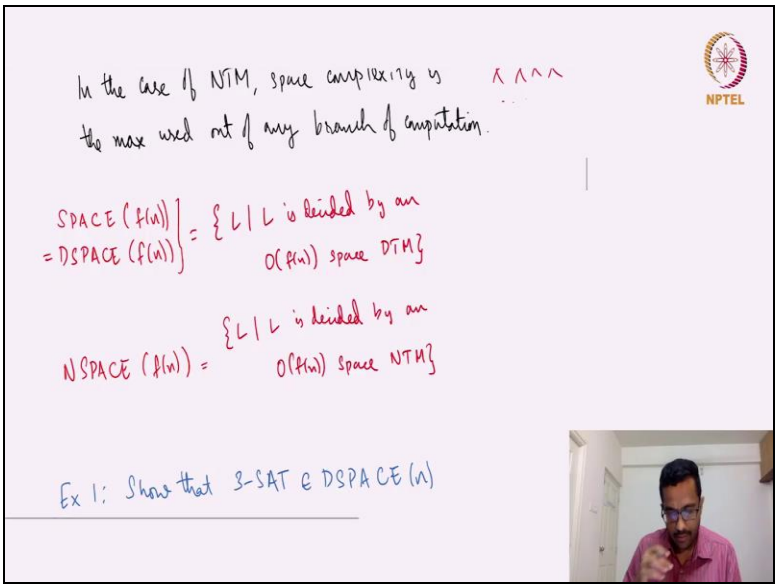
So, these are all yes or no problems but it is not problems like what is the square of this matrix or what is the square root of this number what is the length of the shortest path from a to b in this graph. So, these are all asking us to compute a certain quantity. So, these require an output type if it is a decision problem we will not need an output type. So, only when output tape is required only when you compute f certain function and further the input tape will be read only.

So, the input will take n symbols and that is read only. So, you cannot really the machine cannot use that space to do the computation. So, you cannot cheat using that space and the work tape you can do anything there you can read you can write you can erase you can overwrite all of this and the output type if it is a it is a computing of a function this will be only. So, just because the output is happens to be very long we do not end up penalizing the machine.

So if the output is long if you if you have to write a matrix product that itself sticks takes order n squared space. So, n by n matrix just to write down union n square symbols at least n squared symbols. So, but maybe the actual computation itself does not require that much. So, just because output is long we do not want to penalise the machine. So, what will really be will be accounted for is the work tape.

So, this is the space that counts we do not care about read on the input tape we do not care about the output type if there is an output the only thing that we consider is the work type usage.

(Refer Slide Time: 08:55)



So, space complexity of a turing machine is the space used by the workday. So, now we could have work the space usage being sub-linear like it could be log n and still we could count that the space usage in the work tape in the as a function of n. So, just like time complexity. So, I am not formally writing down the definition of course this is. So, for this part also you can follow Sipser chapter 8 there is a chapter on space complexity you can follow that. I am pretty much following that only.

Maybe except for some change of ordering in the in the way topics are presented. So, the space complexity of the space use is a function of n. Meaning if for over all the length inputs of length n what is the maximum space usage for a certain computation. In the case of; so, this is in the case of a deterministic turing machine. In the case of a non-deterministic turing machine we look at again just like in the case of time bounded computation we look at all the we look at all the computation paths.

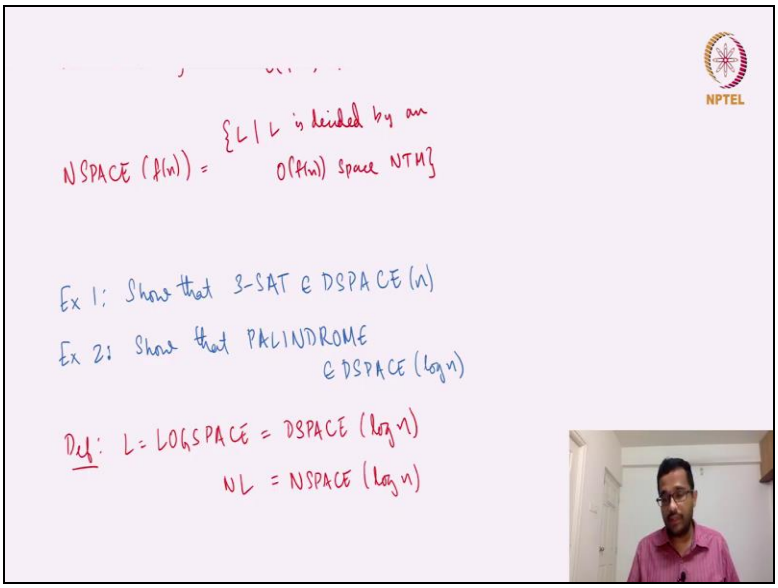
So, all the computation paths and out of all the computational paths on all the inputs of length n which uses the most space that is the space complexity. So, space complexity in the case of deterministic tuning machine is just the maximum amongst all the inputs of length n in the case of non-deterministic machine we also need to account for the multiple branches of computation. So, now let me just formally define this notation this is like analog of D time t n. So, we said D

time t n is equal to all that can be computed with time complexity t n with a deterministic turing machine.

So, similarly space of fn or space of S n is it of all languages L where L can be decided by an order fn space deterministic turing machine. So, but if L is decided by an deterministic tuning machine in order fn space then L belongs to space fn and N space fn is the same thing it is analog of n time by N order fn space DTM. So, the only thing that changes is that D changes to n. So, if m if a language can be decided by an order fn space NTM then it belongs to N space fn otherwise if it is decided by order fn space DTM it is called it is in D space fn.

And we will not be using the D before D space fn sometimes we will just drop the D and say S space of fn. So, now I have formally defined what is space complexity and the complexity class of space of fn and n space of fn.

(Refer Slide Time: 12:16)



Now some simple exercises I can just briefly give you an idea of how to how to go about attempting these problems. So, we saw three SAT it is an NP complete language and it is entry complete means it is one of the hardest problems in the class NP. Surprisingly it requires only linear space and deterministics linear space. So, the thing is that. So, how do you how can you show that you could have a counter let us say there are n variables x 1 to x n.

So, there are 2 power and possible assignments. So, you could have a counter saying which starts with all false then maybe all. So, all false instead of let us say true falsely let us think in 0, 1. So we you can think of the starting assignments as all 0's then 00001 then 000111 and so on. So, you could just have a binary counter kind of thing and this counter could go through all the possible assignments. So, this counter requires order n space because it is n bits, n variables.

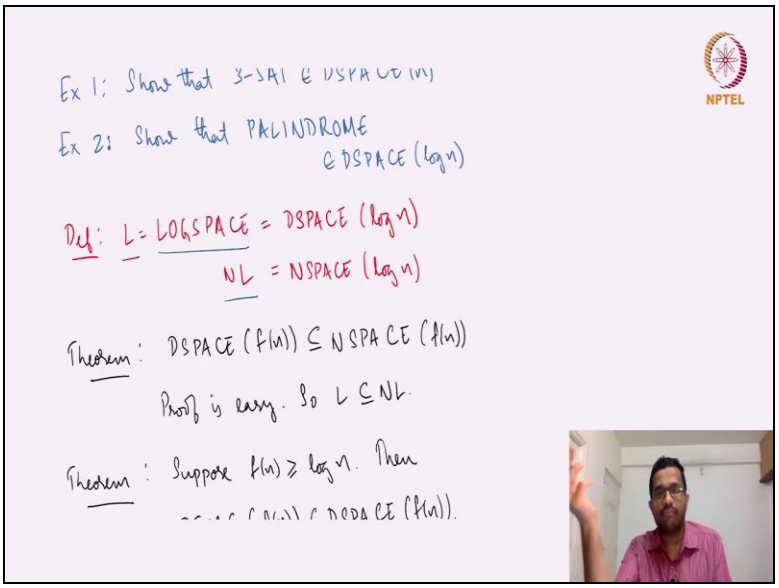
So, you could just cycle through all the possible assignments and once you have an assignment let us say you fix 10101 or something like that. And then plugging in this assignment and testing in the formula whether it is a satisfying assignment or not a satisfying assignment can be done in linear space again this is not. So, hard to see you can work that out. So, satisfiability or at least three SAT is in D space because reset has a certain structure another language.

So, it is palindrome where you have to decide whether the input string is a parent room or not this in order log n space or deterministic space login. So, this is an example of a language that does not even require n space. So, some sub linear space. So, usually N order n is referred to as linear anything that is less than that is referred to as sub linear and that is more than that is referred to as super linear.

So, log n is sub linear why is palindrome in D space n because you could consider the following approach you could just check with the first symbol then the last symbol then the second symbol the second last symbol. So, all you need is to have a counter to keep track of whether you are in symbol I and n minus i or n - i + 1. So, you just need a counter to check to keep track of where you are checking and then you that is the only extra space and maybe one or 2 extra slots. So, here you saw a symbol a. Now in the like the first symbol is a.

So, you also want the last symbol to be a if it has to be a parent rule otherwise you immediately reject. So, palindrome is in deterministic log space.

(Refer Slide Time: 15:31)

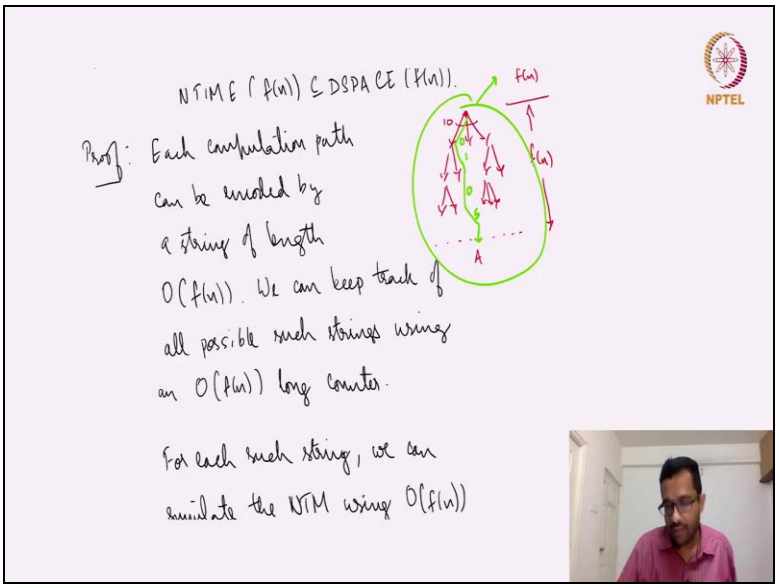


Now one further definition the class the class L or sometimes we call we say L is short for log space is simply D space of log n and the class NL is simply n space of log n. So, L and NL are deterministic log space and non-deterministic log space respectively. So, we saw what is space complexity we saw D space or space of fn and N space of fn and. Now we are seeing L and NL which are specific where fn is equal to log n.

So, now let us see some basic relations between. So, we saw space D space D time n space and we will see some relations between each one of these. So, the first D space of fn is always contained in N space of fn why is this? This is because anything that a deterministic machine can do with a given space a non-deterministic machine can also do in the given space because in order to basic machine simply has more option or more flexibility does not mean that it needs to use that.

So, just like D time if fn was also a subset of n time fn. So, this is the reason and this also tells us that L is contained in n because this is a corollary of this theorem.

(Refer Slide Time: 17:11)



Now let us look at the next result. So, D space, so, this is result number one D space of fn is contained in space of fn. Second is the time of fn is contained in D space of fn meaning if you can do some computation in D time fn you can also do the same computation in D space fn. This may seem a bit challenging why is that true but the reason is very, very simple if something can be computed in D time fn that means in fn many steps or order fn many steps you could have written at most fn cells of the tape.

So, each time step you can only go one, one step. So, at most you would have written order fn cells of the tape. So, the maximum tape usage will be will be at most order fn which means the space required is order fn. So, the proof is that the turing machine again note that both left hand side and hand side are deterministic can write at most one cell of the tape in each time step. So, each time step it can only write one, one cell.

So, the next result is that it relates between end time non-deterministic time and deterministic space. So, now first we saw deterministic time fn is contained in deterministic space f of n. Now we are saying that even non-deterministic time of fn can be contained in deterministic space. So, we know that D times fn is contained in n time fn. So, n time fn is possibly bigger than D time fn but even that is contained in D space fn, why?

So, let us see what we know about end time fn. So, N time fn means there is a turing machine there are multiple possibly multiple possible computation paths right. So, and we know what all that we know is that all the computation paths terminate before tn steps or fn steps. So, this is what we know this computation t the length of the longest path from the root to the leaf is fn. Now the point is this we may we can assume that at each the branching factor of the turing machine say the branching factor is a constant let us say it is 10.

So, because the branching factor only depends on the only depends on the turing machine and not the length of the input. So, let us say the branching factor is 10. So, now we can encode the path each path that it takes. So, let us take this path some path like this. So, you are saying that first I took the zeroth branch then the first branch then the second branch or maybe not second may be again the zeroth branch and then maybe the fifth branch and so on.

So, I can write the identity of this path as 0, 1, 0, 5 and so on because I first took branch 0 then 1 then 0 then 5 and. So, on where this string is of length fn. So, each path that the machine takes can be encoded by an fn length string we can keep track of. So, now all that we need to do is to. So, you have a non-deterministic turing machine or non-deterministic turing machine that runs in fn time we need to; in a deterministic sense how do we how do we simulate this or how do we check all the paths, because we need to simulate it with a deterministic space steering machine.

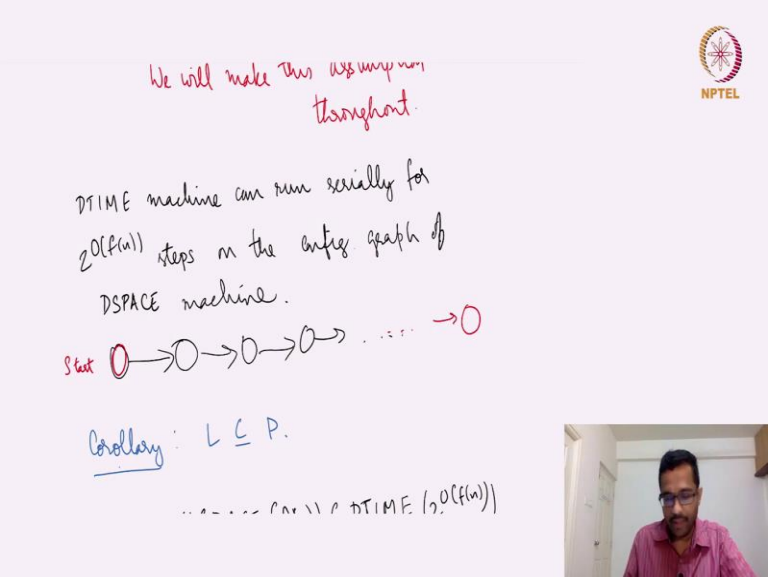
So, what we can do is we can we can have a counter. So, here we are saying 0105 is the identity of this path. So, you can have a counter that generates all the length fn sequences 00000 there is a 10023 so on. So, these are all possible paths and for each of these paths the machine can come execute this computation. So it can check for if it takes zeroth path first part zeroth path fifth zeroth branch first branch zero trans fifth branch so, on does it lead to accept or not and if any of these parts accept then the machine also accepts.

So, the deterministic simulation is simply check out all these paths all these different branches. So, you basically do an exhaustive search in the in this computation tree and this does not require more space than order fn because you are not maintaining the entire tree you are just checking

each path by path and this each path is encoded by an order fn long string. So, the extra space required is also order fn and each computation is also order fn because it is also order fn time.

So, it is also order reference space. So, N time fn is contained in D space fn non-deterministic time fn is contained in deterministic space fn. So, now what the previous thing that we saw also follows from this D time fn is obviously contained in N time fn. So, it is also contained in D space fn.

(Refer Slide Time: 23:22)



Now what else? Now let us see. So, we saw time complexity classes how is it contained in space complexity classes. Now we will see the opposite how is D space fn contained in the time of D time. So, what D time of what? So, we already know that D time fn is contained in the space fn. So, this same amount of time same function time is contained in space for the same function is space contained in time possibly not because we already saw that sat is in D space n.

So, it is in linear space if we know that this D space fn is contained in D time fn then that would imply that SAT is in D time n which would mean that it is in polynomial time which we know is not between which we do not know as of now. So, it turns out that what we know now is that D space fn is contained in D time 2 power order f n. So, it is an exponential time that much why is this the case? So, D space fn is contained in D time 2 power order fn.

So let us first count the number of configurations of the D space machines. So, D space machine there are multiple components to the configuration. So, one is the input is fixed. So, the input is read only on also. So, the input cannot change but the work tape has fn space and each of this fn space could contain any symbol and let us say sigma is a set of same alphabet. So, size of sigma is a number of symbols so there are sigma power fn based to fill up the work tape multiplied by there are 2 heads one in the input tape and one on the work tape.

So, the input tape has n locations so, because it is only the input. So, the head could be in any one of the n locations and the work tape has fn locations. So, the head could be could be anywhere in the fn locations. Finally the last part of the configuration is the state. So, let us say Q is a set of state. So, size of q. Now let us say this is. So, now the sigma I can write as 2 power c 1 for some constant c 1 2 power c 1 fn multiplied by n times fn times size of Q size of Q is also a constant number of states is a constant.

Now I can simply write this as this is less than or equal to power c 2 of fn because n is. So, suppose fn is at least log n then n can be written n m n is dominated by 2 power fn and fn is also dominated by 2 power fn and Q is a constant. So, that is also dominant way to to power fn. So, all we need to do is to change the constant here from c 1 to c 2 slightly higher constant. So, the number of configurations of the D space machine is 2 power order fn.

So, this is 2 power order fn sorry order fn configurations that those are the number of configurations of the D space fn machine. Again here we assume that fn is log n at least log n and this is assumption that is helpful to make throughout this throughout this part not this lecture but this part where we talk about space complexity because just to read the input in an in a reasonable manner.

So, the input is of length n just to read the input we will require the need of some counter which keeps track of where we are and that counter has to be at least of log n bits. So, if you are not even allowed login space the kinds of things that we can do are extremely trivial and they are not very interesting. So, throughout we will assume in the case of space complexity that fn is at least log n and it is because this assumption is not penalizing us much.

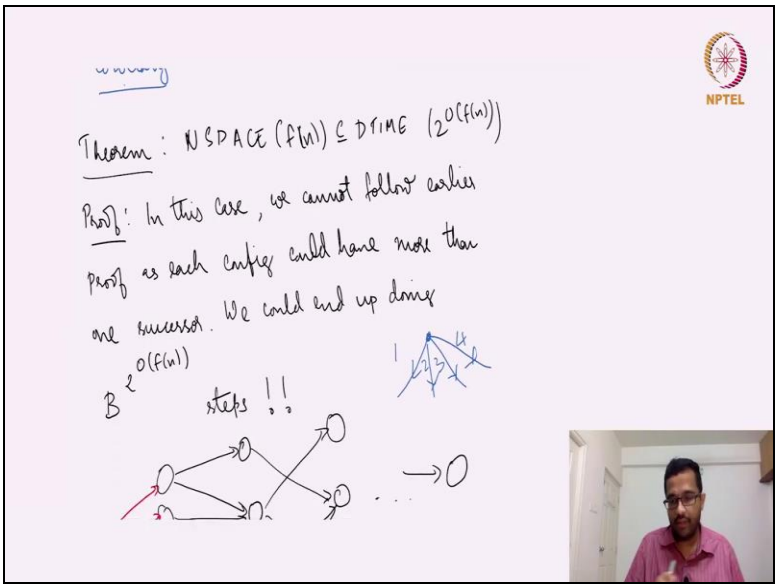
When this is not true then the classes that we have are not very interesting. So, we'll keep this as an assumption throughout. So, now we know that there are 2 power order fn configurations. So, now let us say there is a starting configuration let us say this is the starting configuration this red one this is a starting configuration it is a deterministic space bound machine. So, it followed by another configuration it is followed by another configuration and we need to keep going like this till we hit accept or reject the computation ends when you reach accept or reject.

But how many steps should this computation go? So, re note that there is another possibility instead of accept or reject it may come back to some some earlier configuration then could just keep looping. But but how do we check that? Again we probably cannot check that but what we can do is to count the number of steps that we have done. How many configurations that we have crossed this is because we know the number of configurations is bounded by 2 power order fn or 2 power c 2 fn.

And if it is looping we will just terminate after 2 power c 2 fn because then it is never going to come out of it. If it loops it has to loop before 2 power c 2 fn. So, because there are no 2 power c 2 configuration. So, if it comes back to another configuration it will happen or if you have cross 2 power c 2 fn time steps and you are still not done that means you are you are looping you have come back to some earlier configuration all.

So, all you do is you just run the turing machine up to 2 power order fn steps. So, instead of order fn maybe you can make it more precise to power c 2 fn or whatever on the configuration graph of the D space machine. And if it accepts by then good if it does not accept by then that means it is not going to accept any longer if it rejects by then that is also fine. So, this simulation will take at most 2 power order reference steps that is why deterministic space of fn is contained in deterministic time 2 power order fn.

(Refer Slide Time: 30:43)



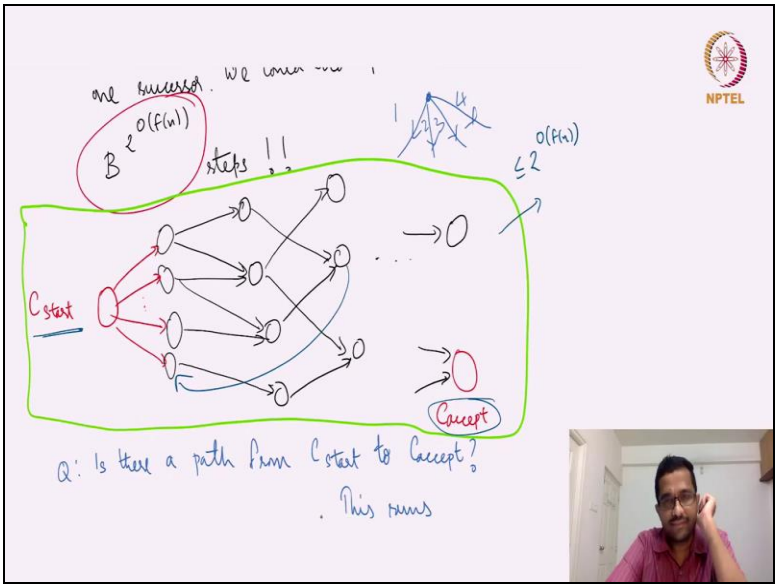
Again when fn is at least log n and we will throughout be making this assumption again. Now if we set fn to be log n we get that L is the left hand side is 2 power sorry D space fn. So, D space log n which is log space L and the hand side is 2 power constant times log n D times 2 power constant times log n which is which corresponds to P because 2 power some constant times log n this is nothing but n power some constant.

So, D time 2 power constant log n is simply D time n power constant which is the class p, so, we saw D space and n space the time how is it contained in this D space D time how is its container N space then we saw D space how it is contained in sorry D space how it is contained in the time. Now we will see N space how is it contained in D time. So, N space is potentially bigger than D space because we already saw D space is contained in n space turns out even n space of fn is contained in the time of 2 power order fn.

So, let me just fix my handwriting a bit D time of 2 power order fn. So, this is the same hand side that we had for D space as well. So, it turns out that for n space also the same right hand side suffices in this case. So, again the configuration bound is the same because the configuration the number of configurations does not depend on whether it is a deterministic turing machine or a non-deterministic turing machine all it is, is a snapshot of where the turing machine is now.

It is not the something that is indicative of the dynamic movements of the turing machine it is just a snapshot of where the turing machine is now. So, the number of configurations is 2 power order fn. But however in the case of deterministic time machine or deterministic space machine we know that every configuration has a single unique successor. Starting configuration gives rise to c 1 and c 1 gives us c 2 or some something like that it goes like that and so, we just need to execute that.

(Refer Slide Time: 33:40)



In the case of N space machine we could have many possible like we could have a branching factor the starting configuration could have 10 possible children successor configurations. The each successor configuration could have its own 10 possible successors. So, there is a lot of branching possible. So, suppose the branching factor is B for each state each configuration you could have b successor configurations then the total number of like the paths that we have to explore could be as large as b to the power 2 to the power order fn which is something humongous.

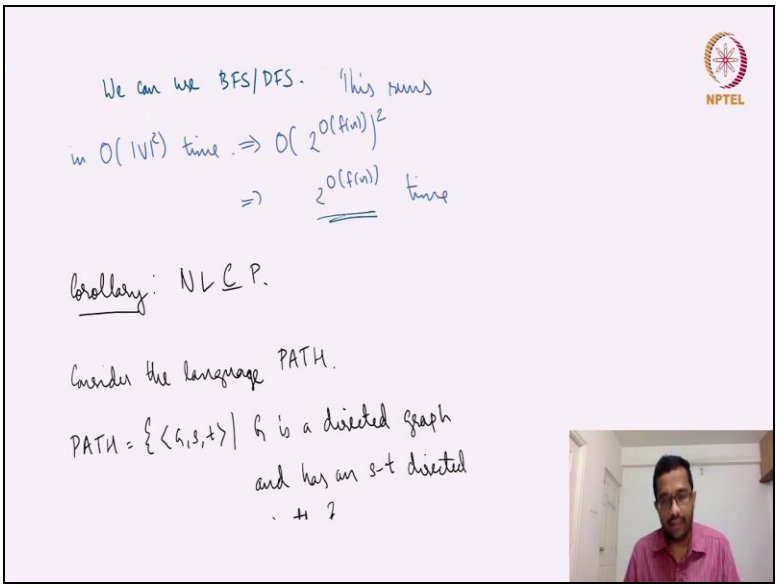
So, this is not something that is desirable and our right hand side we just have 2 power order fn not there is no B there. So, why is that? That is because the actual number of configurations is simply 2 power order fn just that in a in a if you have 2 power B power 2 power order fn computation paths it must be the case that many of these paths are reusing the same configuration. So, the actual number of configurations is only 2 power order fn.

So, but then many times you are probably jumping from one path to the other and all these kind of cross criss crossing must be happening. So, now let us try to draw it in a different way where we just have once one in this picture over here in this picture that I am circling with green we draw each configuration only once. We have the starting configuration for a specific input there are a bunch of successors.

And it is possible that 2 successes have the same share a successor it and anything like that and maybe even this kind of things are possible from the second state you may come to the something like this all these things are possible it is a non-deterministic machine. So, this there could be looping on some parts but there may not be looping on other parts etcetera. All we are interested in is there a path from the starting state to the accepting state in this configuration graph.

Is there a path from the starting state to the accepting state in this configuration graph? We know that so, each node each vertex in this configuration graph is the configuration. So, we know that this graph has at most 2 power order fn vertices. And there are 2 designated states or configurations one is the starting state starting configuration and one is the accepting configuration we want to know whether there is a path from the starting configuration to the accepting configuration and this is a directed graph, how do we do this?

(Refer Slide Time: 37:07)



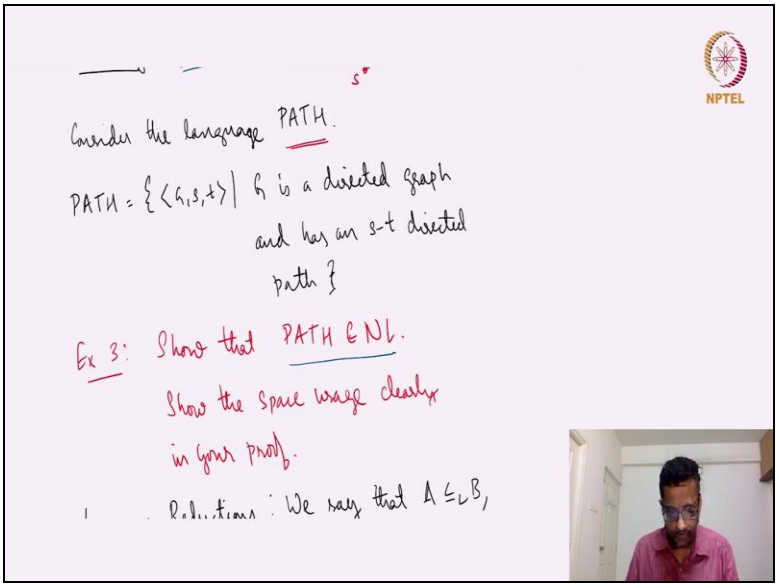
This is something that should be very familiar to you to all of us. So, what we do is we could use any traversal algorithm. So, let us say we use breadth first search or depth first search on this configuration graph. And this conf this runs in order V square time let us say you use the adjacency matrix representation it runs in order V square time where V is the number of vertices of this graph. So, it is 2 power order fn is the number of vertices.

So, it is square of that but again there is an O here. So, it just the square just can be absorbed by the O in the exponent. So, in and in this outside O is kind of pointless it is not required. So, 2 power order fn time. So, that is how even n space of fn is contained in D time of order fn. So, if you plug in fn to be log n, N space of log n is nothing but NL non-deterministic log space and D time of 2 power order fn is nothing but 2 power polynomial it is D time of polynomial which is P.

So, what is the exact problem that we tried here or that we had here? So, you are given a directed graph and there is a specific vertex and there is another specific vertex is there a path from the first vertex to the other second vertex that is that I specified. So, this problem turns out to be somewhat important when studying space complexity and we will call this problem as path. Path is simply given a graph a directed graph and 2 vertices S and t it is asking whether there is a path from S to t a directed path from S to t.

So, is there a sequence of vertices or edges such that you can start from S and you can somehow reach t this is path. So, it is a it is a decision problem and. So, I will stop after this one small exercises.

(Refer Slide Time: 39:34)



Show that path is in NL this is a non-deterministic log space. So, we can only use logarithmic space. So, we cannot you cannot do the breadth first search in the exact same way because you cannot be maintaining a visited marker and all that and we have to be more careful with that. So, show that path is in another. So, just because if you if you maintain visited markers in each node then that is itself order n space required.

So, you cannot you do not have that space. So, show the space usage uh. So, what I am saying is show the space usage in the proof and log space reductions I think we will see it in the next lecture. So, just to summarize what we saw. We saw the space complexity model which is the input type work type and output tape. We formally defined space complexity classes D space of fn and space of fn we said L is D space of log n and NL is n space of log n.

Then we saw relations between the space complexity classes and the time complexity classes the generic classes D space of fn is contained in N space of fn. D time of fn is also contained in the space of fn, D N time of fn is also contained in D space of fn. So, this is time classes contained in

space classes then we saw space classes contained in time classes. So, D space of fn is contained in the time of 2 power order fn classes sorry 2 power order fn ah.

So, if you have the same space or if the time of fn is contained in these space of fn but this space of fn is contained in D time of 2 power order fn this also implies that L is contained in P we already saw P and the non-deterministic equivalent. So, N space of fn is containing D time of 2 power order fn. So, even if you have n space instead of D space even that is considered D time of 2 power order fn which implies that NL is contained in P.

And finally we define the language path and I asked you to see why it is in non-deterministic log space and I think with that we will stop this lecture, thank you.