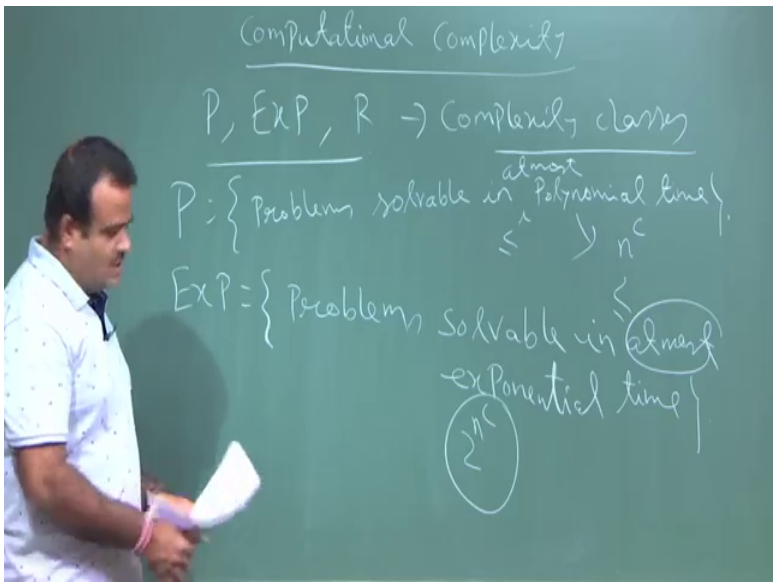
An Introduction to Algorithms
Prof. Sourav Mukhopadhyay
Department of Mathematics
Indian Institute of Technology, Kharagpur

Lecture – 59
Computational Complexity

So we talk about we are almost end of the lecture. So, we talk about comp computational complexity. So, what is feasible, what is not feasible in polynomial time. So, for most of s comma t s comma t s comma t the algorithm we have solved we can we have a polynomial time solution for that. So, we have a. So, now, this class we want to discuss when you cannot do that I mean when you cannot there are many problems which cannot be solved by polynomial times. So, so we will talk about complexity classes.

(Refer Slide Time: 01:01)



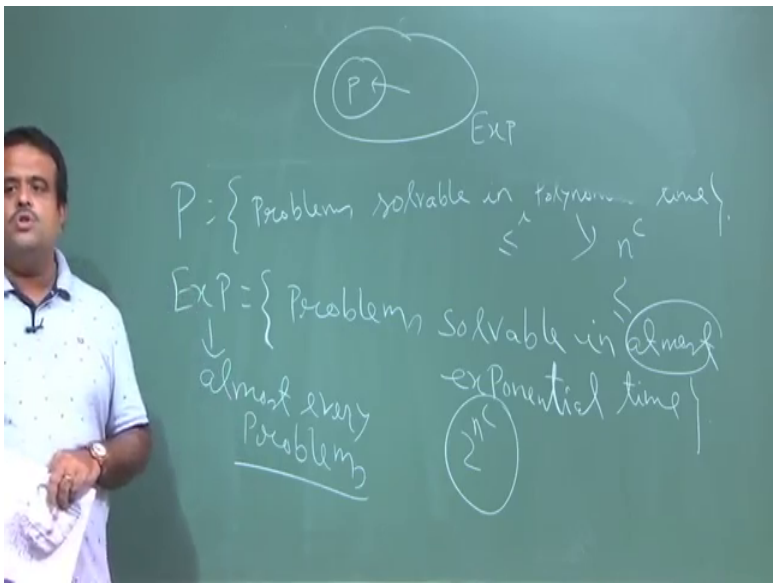
So, basically we talk about three complexity classes P x and i. So, these are basically complexity classes.

So, what is P? P is basically the set of all problems which are solvable in polynomial time. So, it will take at most polynomial time. So, P is basically problems which are solvable in at most polynomial time in polynomial time. So, polynomial time means the complexity is sum n to the power c where c is a constant. So, this is less than equal to at most. So, at most polynomial time, less than equal to, ok.

So, this is the class where we consider all the problems which can be solved in polynomial time. We have seen many problem so far in our lecture. So, most of them we have seen solved in polynomial time, but today we want to know there are a lot of problem which cannot be solved in polynomial time. So, the second class is x, which is basically problem solving in exponential time problems solvable in at most that less than equal to exponential time that is why exp x; exponential time. So, this is at most, less than equal to. So, this is the class where we put all the problems, which is which will take exponential time.

So, exponential time in particular we look at 2 to the power n to the power c time. So, here n is a constant. So, this is the time complexity for such a problem. So, for example, so all the polynomial times here algorithm is subset of this. So, if we want to draw a picture. So, if we want to draw a picture. So, almost all problems are belongs to in this class.

(Refer Slide Time: 03:53)

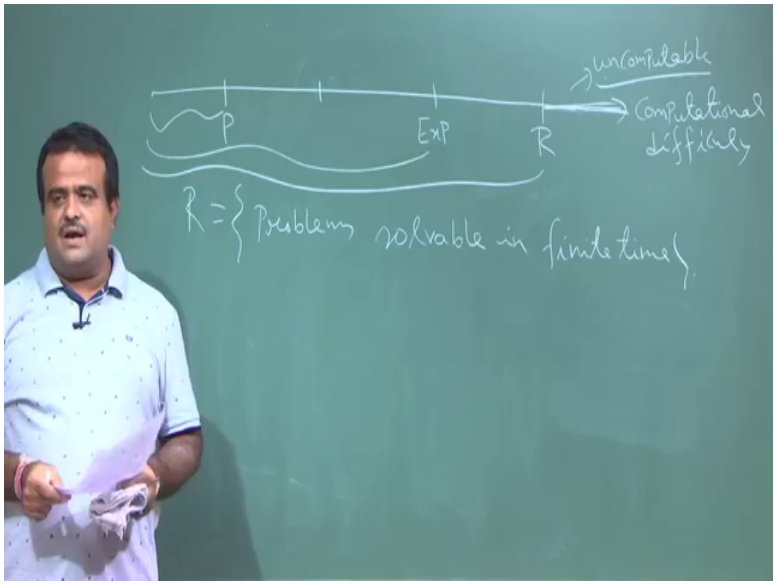


So, almost every problem problems are in this class. So, if you want to draw a picture here this is a P and this is x exponential time algorithm now what do you want we want to bringah one the problem from here to here. So, that we did for say dynamic programming problem, we know this is exponential time algorithm, but we try to solve it in polynomial time by the help of memorization. So, all this technique we use. So, this is

the this is our we want to bring the problem, which we now it is an exponential time algorithm you want to bring it here.

So, P P is basically. So, P is a subset of x we have a other picture to draw that this is not a very good picture to explain this. So, there are another class. So, we want to draw it in different way.

(Refer Slide Time: 05:12)



So, in a line like this, this is basically we denote by computational difficulty computational difficulties. So, here up to this is p; that means, these are easy problem which can be solvable by polynomial time algorithm polynomial time and then we have another one which will explain and we have x exponential time, and then we have here what is called R, R is another complexity class. So, R is we will explain R. So, what is R? R is basically problem which we are solvable in finite time.

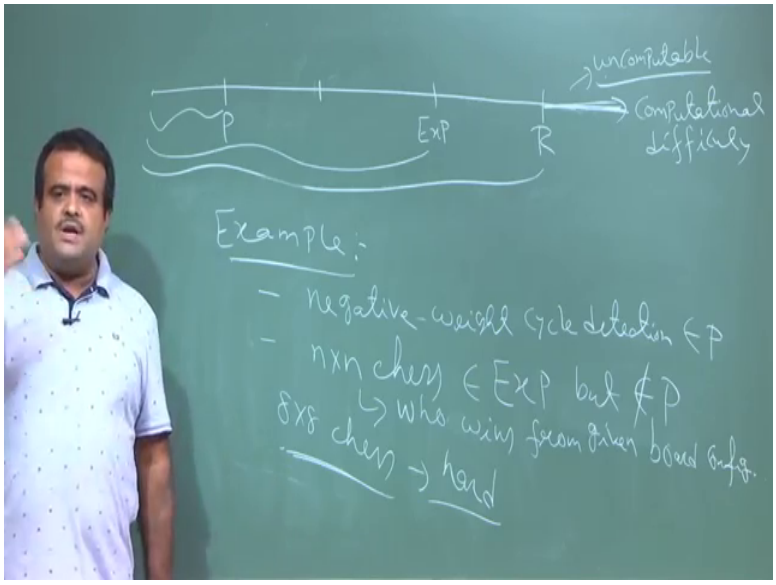
So, problems, this is the set of all problems which are solvable in finite time. So, this is basically R set. So, s R stands for this problem R is the collection of all problems which can be solved in finite time. So, this set is P which is easy problem, this set is x P which is exponential time algorithm and this set is basically R. So, which are the problem which are which can be solving finite time.

I mean. So, we should have a finite machine which can give as a halt solve state. So, anyway i am not going to that theoretical details and there are other problems which are

here. So, these are basically problems which cannot be solved in finite time. So, this will take in finite time. So, these are called uncomputable problem or undecidable problem. So, these are basically uncomputable problem because this problem we cannot get a solution of this problem in a finite time.

So, this problem is undecidable. So, this problem has no solution in a finite time. In fact, most of the problem are in this range we will prove that most of the problem are in this range which are uncomfortable. Now let us take some example of this classes example of problems which belongs to this classes.

(Refer Slide Time: 08:29)



So, let us take example of this classes. So, we want to take an example of polynomial time algorithm we have seen many one this is a negative weight cycle, to determine. So, we have given a graph, we want to determine whether there is a negative way cycle or not. So, this is basically negative weight cycle detection.

So, we know this is in P because we know an algorithm which is bellman ford algorithm which time complexity is order of v into e.

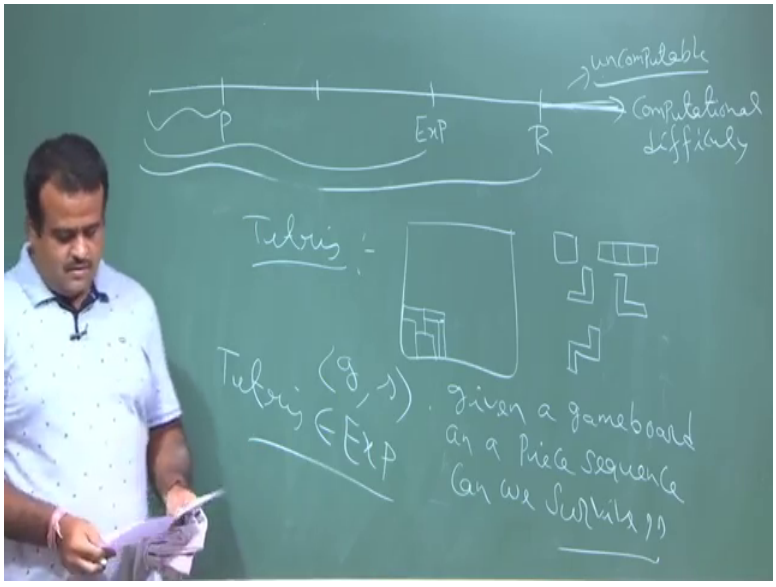
So, that is polynomial time algorithm. So, this is a example of P, I mean a problem which is polynomial time. Now we have to have an example of exponential time algorithm that is n cross n chase problem this is belongs to x exponential time algorithm what is this? This is basically we have given a position of a chess. So, we are playing chess.

So, we have given a position of a chess position of this chess now we want to decide whether quite we not. So, this is the problem. So, this problem has no polynomial time algorithm to solve this problem. So, this problem is exponential time. So, we can try for all possible move and then. So, this is basically exponential time algorithm exponential time problem. So, this belongs to this.

So, the problem is given a. So, who win white like white or black who wins from given chessboard configuration given both configurations. So, we have given a position of the board, now we want to we want to take the decision where it why whether white win or not. So, this is the problem. So, this problem is hard even this is hard for 8 cross 8 cross chess board this is even hard for 8 cross 8 chess box.

So; that means, there is no polynomial time algorithm to solve this problem. So, this problem is hard problem. So, this is basically belongs to exponent a exponential class and it is not in P, even this 8 bindings chess also this is not in p; that means, we are not having a polynomial time algorithm to solve this problem. So, this is a typical this is an example of exponential time algorithm. So, another exponential time algorithm is tetris.

(Refer Slide Time: 11:52)



Another exponential time problem which is basically a tetris what is tetris? We have a board like this is a game. So, we have a board this is a computer game and we have some blocks. So, blocks are of this Form say this type maybe this time this type this type of thing we may have these type blog we may have these type blocks.

So, these are the types of blocks we have seen this is also on type of blocks. So, there are all possible blocks and then these blocks are falling and we have suppose we have given some situation like this say like this like this. So, we have given some position of the board and. So, we have given the board and we have given a sequence of blocks.

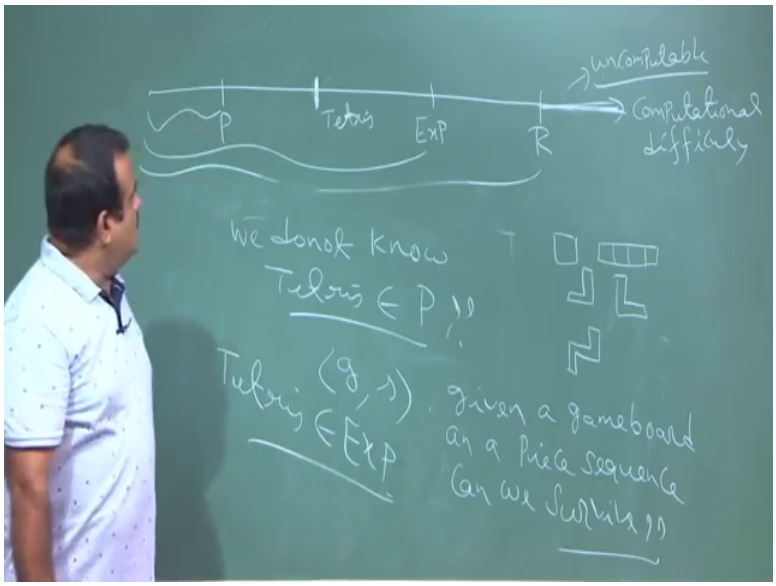
Now we have to decide whether we will survive or not, whether we will stuck or not either we will survive or not. So, if there is a matching then this will have this something like that. So, when we will not survive, if it is starts the top of the board. So, the problem is tetris problem is. So, given this g comma s, s is basically a sequence of these blocks. So, given a board given a game board and a piece of sequence, I mean we have a this blocks, we have these shapes can we survive? This is the question this is the problem.

So, can you survive so; that means, can we just this this is coming like this. So, it will fall like this. So, if it is if this row one row is complete then it will just vanish like this, so this game.

We have seen this is a computer game this is called tetris game. So, this is hard. So, there is no polynomial time algorithm to solve this, to answer this question whether we survive or not. So, this is hard.

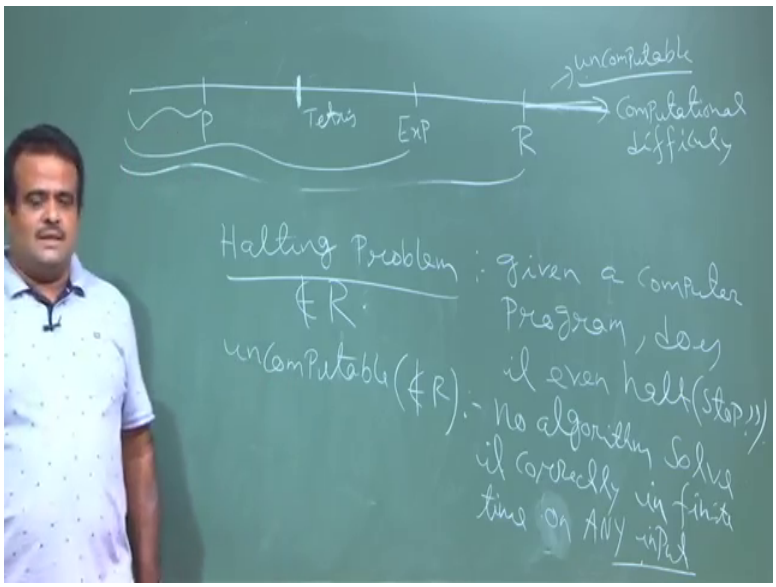
So, tetris says this is hard. So, this is another example, but we do not know whether this is belongs to P or not I mean this is this has no I mean we do not know whether this is belongs to P p or not there is no algorithm. So, far whether this can check that given this whether we can survive, that is we do not know whether.

(Refer Slide Time: 15:11)



So, this we do not know this is a question mark. So, so this is another example of the exponential time algorithm. So, tetris is suppose here tetris. So, we do not know I mean this is exponential time algorithm, but we will see I mean it is also n we have not defined np we will talk about np so. So, now, let us talk about R some example of the problems which are say not in R say for example, halting problem.

(Refer Slide Time: 16:06)



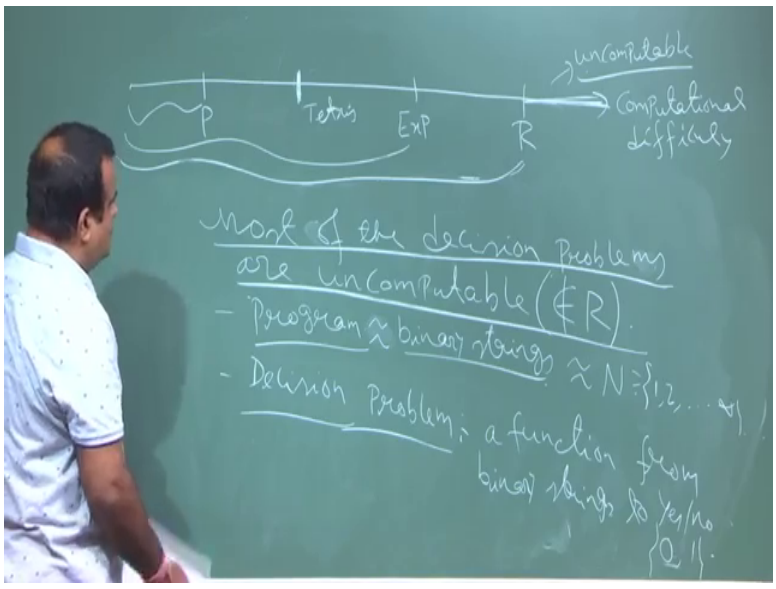
So, what is a halting problem? So, you have given a computer program and the question is does it ever halt. So, given a computer program the question is does it ever halt stop

actually; that means, stop. So, this is the problem, problem is we have given a computer program the question is does it ever halt does it ever stop. So, this has no finite there is no algorithm to solve this problem in finite time for all input there may be for certain input we can say the answer is yes or no, but in general for all input it is there is no such algorithm, we can say that the answer of this question. So, that is why this does not belongs to R so; that means, there is no algorithm which can tell given a arbitrary program whether it will halt or not in a finite number of in in a finite time.

Obviously, in infinite time we can say because we can run the code and it may go up to infinite time. So, at infinite time we have the answer whether it will, but for finite because for infinity we can run the. So, we can run the code, we can run the program and then we can see so, but there is no program. So, this is basically uncomputable computable because this is not in R so; that means, no algorithm there is no algorithm solve it correctly infinite time for any input any. There may be certain input we can say yes, but in general for an arbitrary input there is no algorithm which can say that. So, this problem is not belongs to R. So, this problem is uncomputable problem. So, this this problem may take in general infinite time.

So, now our claim is most of the decision problem is uncomputable, most of the decision problem does not belongs to R so.

(Refer Slide Time: 19:46)



So, this is our next conclusion, so most of the decision problem. So, what is decision problem will explain. So, where the like problem we have discussed what is that in the negative weight cycle. So, this answer is yes or no. So, this is decision problem this program will give an program computer program whether it will halt or not yes or no. So, this has this is called decision problem. So, you have two answer yes or no. So, every problem has a decision version of it, most of the problem has its decision version. So, most of the decision problems are uncomputable; that means, not in R; that means, not in R most of the decision problems a lot in R ok.

So, what is the decision problem? Decision problem means the answer is yes or no so; that means, does this program halt yes or no. So, given a given a board play board and a sequence of this sequence in tetris whether we can survive yes or no. So, given a position of the chess configure given a position of the chess board whether white wins or not. So, this is all our decision problem more. So, every problem most of the problem has a decision version of it. So, we want to see why this is true I mean almost all of the decision problem is does not belongs to R.

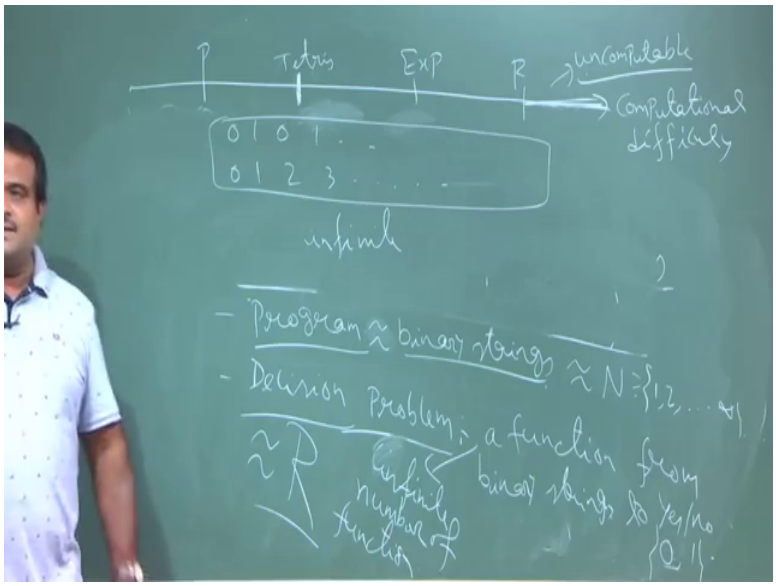
So, why it is true? Because, if it is in say if we can be solve in finite machine. So, there is a program, computer program. So, program basically. So, program either in C C plus plus Python anything, but ultimately it will convert into ASCII and ultimately it will convert into machine code. So, machine code is basically zero on bits because computer can understand only the 0 1 bits. So, it is ultimately converted into 0 1 bits. So, basically it is a binary string. So, this is basically a binary string. So, every program is ultimately converting into binary string.

Now, if you have a binary string 0 1 bits. So, that is basically a natural number. So, by every binary string we can think of a natural number set of natural number natural number means 1 2 3 4 this set this is basically said 1 2 3 4 up to infinity natural number. So, every binary string can be converted into natural number. So, every program is a basically converting into a natural number. So, if we have a program it is basically a natural number there is a one to one correspondence. Now we would like to see the decision problem how we can see the decision problem. So, decision problem is basically it is a function from binary string to 0 1 because we have a this we have a problem we have to say yes or no. So, the it is a basically a function form binary string to yes or no;

that means, yes or no; that means, 0 or 1 we can say yes is 0 like this this is our convention.

So, this is a function form a binary string to yes or no now how many such function is possible. So, that we have to see. So, how many such functions are possible.

(Refer Slide Time: 24:38)

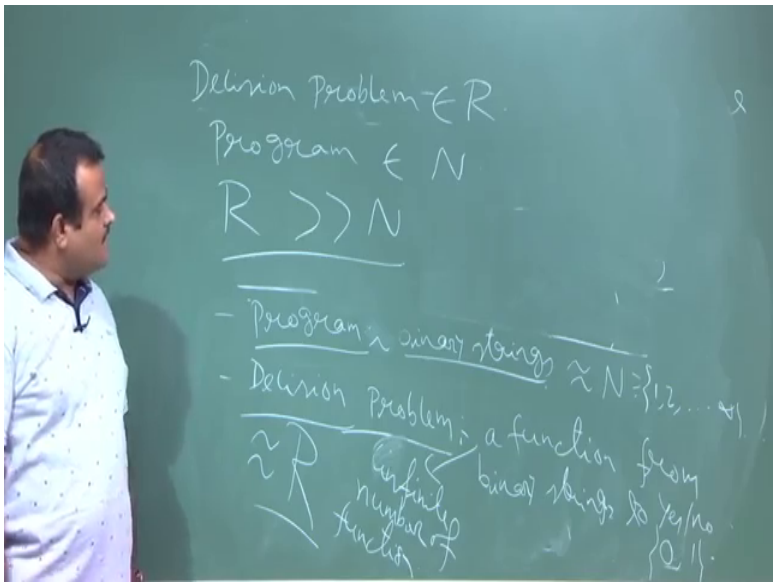


So, basically we will keep this, this is R. So, so basically we have given a string and then we are converting into binary. So, this is a function form a binary string to 0 1. So, how many such function is possible? In a other word we can say we can consider this as a stable. So, we have all possible binary string binary string can be mapping from to the natural number, these are all possible binary string then we have a say 0 1 0 this is one functions.

So, how many such function are possible? If this string is finite then it is if there are

N bits and 2 to the power n because each of the field can be field by n ways like two ways. So, 2 to the power n, but there are n possibilities I mean this is capital N. So, there are infinite number of bits. So, this is basically infinite size is infinite. So, there are such function is infinite uncountable infinite number of such functions are possible. So, this is this is basically R set, set of all possible real numbers. So, this is a uncountable remaining infinite set.

(Refer Slide Time: 26:29)



So, our decision problem is belongs to R set and our program computer program is belongs to N set. So, now, R is far greater greater than N so; that means, we have very less number of program for our soul, because each program will give a solution of a particular problem and we have uncountably many problems and we have only countably many programs exist. So, there are many problems which are unsolvable, because this set is very much bigger than this set. So, there are many problems which are basically unsolvable.

So, this is the proof sort of where this is basically telling us the decision problem is uncountable. So, this is the bad news. So, there are many problems I mean, there are many problems which cannot be solved by the computer program and; that means, that is. So, there is no finite number of step we can solve this. So, these are uncountable these are beyond R because our computer problem is basically a given problem can solve one particular given program can solve one particular problem. So, this set is n this set is R this is very big set. So, that is the bad news we have many problems which is unsolvable.

So, we stop here, in next class we will discuss the more notion like np, np hard, np complete. We will continue in the next class.

Thank you.