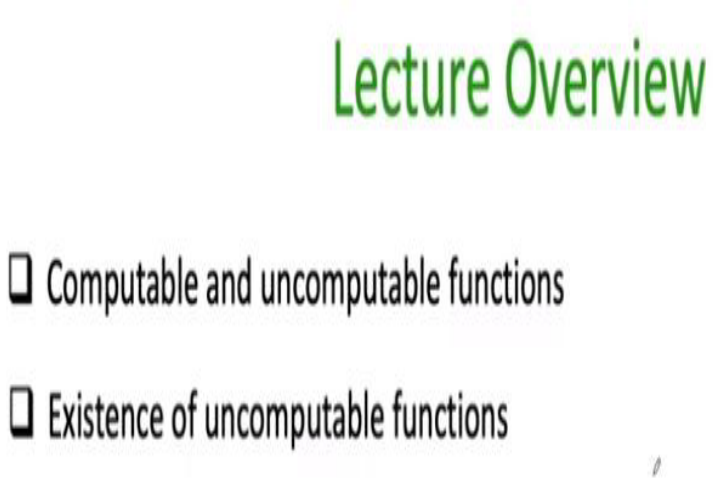
Discrete Mathematics
Prof. Ashish Choudury
IIIT, Bangalore

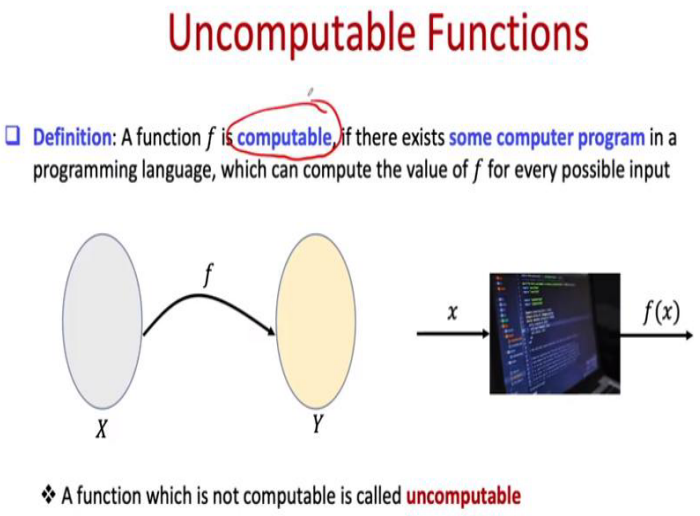
Module No # 06
Lecture No # 30
Uncomputable Functions

(Refer Slide Time: 00:26)



Hello everyone, welcome to this lecture so just a quick recap; in the last lecture we discussed about Cantor's diagonalization argument. And we saw examples of uncountable sets; the plan for this lecture is as follows. We will discuss about computable and uncomputable functions and we will discuss about the existence of uncomputable functions.

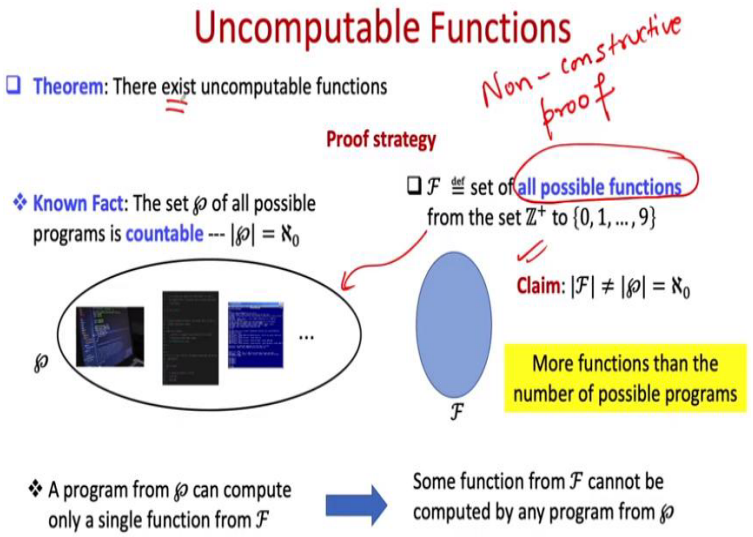
(Refer Slide Time: 00:47)



So what exactly are uncomputable functions; so they are some special type of functions. So say I have function f defined from a set X to a set Y then I will call the function f to be computable if there exists some computer program in a programming language which can compute or give you the value of this function for every possible input from the domain of that function. So mind it I am not focusing here on the running time of the computer program or the resources utilized for the program to give you the value or the output of that function.

I am interested whether there exists a program or not which can give you can give you the output of that function for every input from the domain. If you can write a program in the programming language for such a function I will call that function to be a computable function. Otherwise I will call it uncomputable function. So as per the above definition a function which is not computable will be called an uncomputable function.

(Refer Slide Time: 02:00)



So what we now want to prove is that there indeed exists uncomputable function that means it does not matter how much resources time and memory space you provide. And write down a program there always exist some function such that you cannot write down a program respective of resources allowed to compute the output of that function for every possible input. And what will be proof strategy that we will follow to prove this theorem?

So we will begin with some known fact; so just recall that in one of our earlier lectures we proved that the set of all valid programs in any programming language is countable. That means we can enumerate them even though they are infinitely many valid programs. When I say programs I mean to say the valid programs; which complies and give you an output. That means it has a begin instruction and an end instruction and a sequence of arbitrary number of instructions in between the begin and end instructions and it complies and it gives you an output.

So the collection of such programs is denoted by the set $\mathcal{P}$ calligraphic $\mathcal{P}$, so in one our earlier lectures we proved that even though we can have infinitely many programs we can always enumerate them. Namely the cardinality of the set of all valid programs in a programming language is $\aleph_0$. So this is a known fact. What we will prove is we will prove that the set of all possible functions from the set of positive integers to the set of integers $\{0,\ldots,9\}$ call that set to be calligraphic set $\mathcal{F}$.

We will prove that the cardinality of this collection of all possible functions is not $\aleph_0$ that is what we are going to prove. That means what we are going to show is that we have more functions than the number of possible programs. Because the number of possible programs is $\aleph_0$ but we will show that we will have more number of functions from the set of positive integers to the integers {0,..,9}.

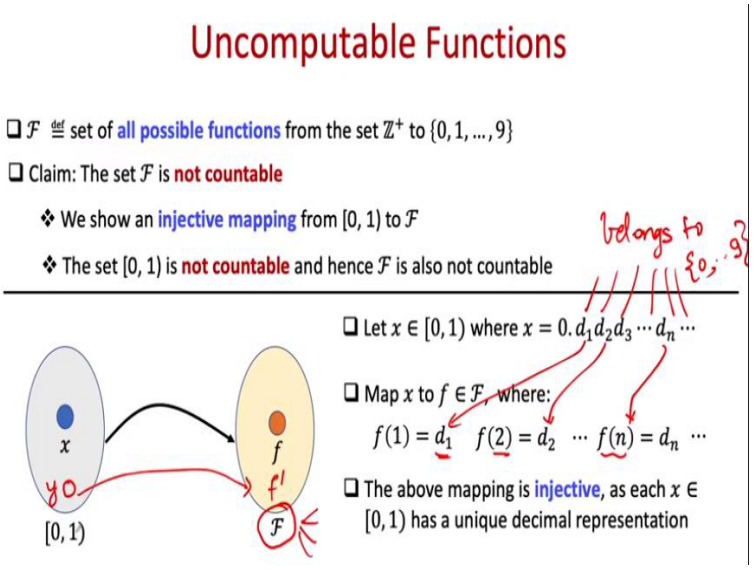
Now any program from your collection of valid programs can compute a single function from this collection F. We cannot have the same program which gives you the value of both, function f_1 as well as function f_2. Because function f_1 and function f_2 will have different characteristics how can it be possible that you have a common program P_1 which simultaneously gives you the output of function f_1 as well as it gives you the function output for f_2.

You cannot have such special programs; that means if we prove this claim then based on the known fact we come to the conclusion that you have some function in this collection of all possible functions for which you cannot find a matching program in the list of all valid programs in your programming language. That means, there is no program in programing language which can help you to compute the output of that specific function and that is the specific function will be an uncomputable function.

So what is the proof strategy we are using here we are actually arguing about; we are giving here a non-constructive proof. Just to recall what is a non-constructive proof? Non-constructive proofs are used for proving existentially quantified statements. So this statement is an existentially quantified statement because it says that there exists at least one uncomputable function.

And we are logically arguing that indeed one such function exist we are not giving a concerte function for which you can never write a function. We are logically arguing the existence of such a function so that is why this is a non-constructive proof here.

(Refer Slide Time: 06:09)



So the set calligraphic $\mathcal{F}$ denotes as I said the set of all possible functions from the set of positive integers to the set {0,…,9}. And my claim is that this set is not countable that means its cardinality is not $\aleph_0$ or in other words you cannot enumerate out, list down all the functions in this collection. How we are going to prove this? Well we already know a set which is not countable. What is the set?

This is the set of all real numbers between [0,1) including 0 that is why you have the square bracket within 0 but excluding one. So this set is already known to be uncountable what we will show is, we will show an injective mapping from this set to the collection calligraphic $\mathcal{F}$ which will prove that the cardinality of this set of all real numbers between 0 and 1 is strictly less than equal to the cardinality of the set of all possible functions.

And since this collection is uncountable any set which has more cardinality than that collection also will be uncountable. So how do we show the existence of the injective mapping? The mapping is very simple you take any real number x between 0 and 1 possibly including 0. So it will have a decimal representation; let the digits in decimal representation d_1, d_2, d_3, d_n and so on.

And again I am assuming here that the number of digits in the decimal representation is infinite why is that? Because even if x as finite number of digits in its decimal representation say for instance x is equal to 0.25 I can always interpret that 0.25 to be 0.2500; I can always plugin

infinite number of zero's at the end. So that way I am assuming here that the number of digits in the decimal representation of every element x of the domain here is infinite.

Now I have to tell you the corresponding image for this element x, the image is computed as follows and remember the image will be a function. Because this set calligraphic $\mathcal{F}$ denotes the set of all possible functions from the set of positive integers to the set $\{0,\ldots,9\}$. So it will be this set is basically a collection of functions so I have to show you one function from this collection which will be the mapping of this element x.

So what will be that function? The corresponding function f which is the mapping of this element x is as follows. The function f(1) will take the value d_1 the function f(2) will take the value d_2 and like that the function f(n) will take the value d_n and so on. So what basically I am doing here is that I am focusing on the function which gives basically the nth digit in the decimal representation of your real number x.

That is the function f here, and since I am assuming that I have infinite number of digits in decimal representation of x this function will take the values and input 1, 2, n up to infinity. And possible values or outputs of these functions can be only between 0 to 9 because each of the decimal digits in the decimal representation of x belongs to $\{0,..,9\}$. So that is the function which will be the image for this element x and it is very easy to verify that this mapping is an injective mapping.

Because if you take 2 elements x and y here which are different then their corresponding or decimal representation also will be different. So x will be mapped to f and y will be mapped to a different function f'. You cannot have 2 different numbers x and y getting mapped to the same function f. So that is why clearly this function is an injective function. So we have shown here that indeed the set of all possible functions from the set of positive integers to the set $\{0,\ldots,9\}$ is an uncountable set.

That means if I go back to the previous slide in the proof strategy I have proved my claim that means you have more number of functions than the number of programs which you can write in a programing language. And since each program can give you the output of only a single

function from the set of all possible functions you have more functions that the number of programs.

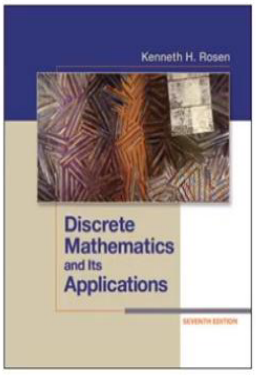
And hence here are some functions for which you do not have a corresponding matching program and that is why you do have uncomputable functions that exist. So that is a very interesting theorem because generally we believe that using computers we can compute anything. You can always write down programs which can give you the output of any function and any computational task in the real world you can think it you can abstract it out in terms of function.

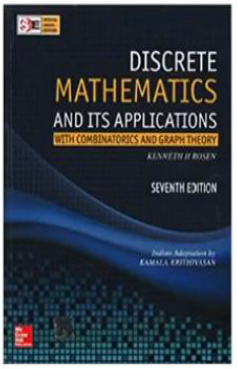
So the remarkable thing about this theorem is that it tells you that its computers are not kind of a god I cannot compute everything using computers or writing programs. There always exist tasks which you cannot compute or cannot compute or cannot find out their values you cannot solve those tasks using computers irrespective of how much time or how much memory you are allowed while writing down the program.

So and all these things we can prove that the theorem proved using the theory that we have developed extensively regarding the cardinality theory. So till now you might be wondering where exactly the concepts that we learnt till now regarding the cardinality theory will be useful. So it is useful while proving the existence of uncomputable functions which is a very fundamental fact in computer science.

(Refer Slide Time: 13:22)

References for Today's Lecture





https://www.anilada.com/courses/15251f16/www/slides/lec5.pdf

So that brings me to the end of this lecture these are the reference for today's lecture. Just to summarize in this lecture we have introduced the notion of computable and uncomputable functions and we showed non-constructively that indeed there exist uncomputable functions which you cannot compute by writing down computer programs thank you.