

Theory Of Computation
Professor Subrahmanyam Kalyanasundaram
Department of Computer Science and Engineering
Indian Institute of Technology, Hyderabad
Lecture 1
An Introduction to The Theory of Computation

(Refer Slide Time: 00:16)

Theory of Computation

One of the most fundamental courses which is part of CSE curriculum (UG curriculum).

Problem 1: I have a computer program and an input x . Will the program terminate when input x is fed?

Problem 2: I have a graph G . Can G be colored using 3 colors (say Red, Green, Blue)

Hello and welcome to the course, Theory of Computation. This will be the first lecture of the course; we will give a brief overview of what we are going to see during the course. So, this will also be a slightly expanded form of the teaser video or the introductory video that we had put out about the course. So, this is Theory of Computation which is one of the most fundamental courses of any B.Tech curriculum in Computer Science.

(Refer Slide Time: 00:48)

using 3 colors (say Red, Green, Blue)

Requires 4 colors!

3-colorable!

Problem 2: I want to drive from home to office. What is the shortest route?

For a computer: one of these is "easy", one of these is "hard" and one is "impossible".

So, let us consider some problems or some questions that are computational in nature. One of them, problem one, is this: I have a computer, I have a computer program, and let us say there is an input that I want to enter into the computer program. And I want to know, if I enter this input into this program, will this program terminate when this input is fed? So, if this input is fed into this program will this program terminate?

So, there are two possibilities. One is that the program runs for a while, terminates, says or gives some output or says accept or reject or whatever. The other possibility is that this particular input somehow leads this program into some kind of an infinite loop situation. So, this is able to figure out some situation where this leads into some infinite loop and it never stops. So, is this such an input that this program will never terminate? So, the question is, will this program terminate when this input is fed into this program? This is one computational question.

The second computation question is: I have a graph G . Graphs are objects like what I have drawn over here. Now, I have a graph, two graphs drawn here. Can a graph be coloured using three colours? So, what is a graph? Graph contains two things, one is vertex, so these dots are vertices. The one on the left has four dots. The lines or the curved lines connecting the dots are what are called edges.

So, the question is about these graphs: can these be coloured using three colours? Let us try for one of these graphs. You want to colour them in three colours in such a way that the nearby neighbouring vertices do not get the same colour. Let us say I start with red for the top vertex over here. These are two graphs, the one on the left and one on the right, so let us first consider the one left. Let us say for the top vertex we gave the colour red.

Now, we have to use another colour, let us say blue, for the one on the right side. Now, notice that the one on the left side now cannot get red because red is adjacent to that. So over here we have to use a third colour, let us say green. Because this green vertex is near to the red as well as near to the blue. So green, we have to use the third colour.

Now, let us see coming to the bottom vertex, we have to get a fourth colour, let us say yellow, because this vertex is adjacent to all the other three vertices. So we need to have four colours to colour this. One may ask the same question for the graph on the right side as well. I am not going to work out, the colourings for the graph on the right side, but the answer is that we can colour it using three colours, so I just say three colourable. This is also a famous graph called the Peterson graph. You can see that it is very nice and symmetric. This is three

colourable whereas for the one on the left, at least the way we tried, we could not get a three colouring.

If you look at the one on the left closely, you can see that whatever you try, you need four colours, because the property that this graph has is that every vertex is near to every other vertex. So the left graph requires four colours, whereas the right one requires only three colours. So this is the question.

So, given a graph, the question is can we colour it using three colours? So, this is another question, you are given a graph and you are asked to colour it using three colours. The third question in the Theory of Computation questions is: suppose I want to drive from my home to the office, what is the shortest route?

Suppose that all the necessary data is given, like the distance from my home to office, which are the different options that I have in front of me like the different road networks, the traffic information, how much time will it take for me for each of these stretches of roads. All of this information is given, can you tell me which is the shortest route?

Now, I have described three problems. First one is, given a program and an input, whether this input will lead the program to termination. Will the program terminate or will it lead to an infinite loop? Second is the three-colourability. We saw two examples as well. The third one is simpler. Given the road Network and the road Network distance or traffic information, can we compute the shortest route from let us say home to office, or it could be any place or any other place, like let us say Bangalore to Hyderabad.

So, we have seen three problems. It so turns out that one of these problems is easy. I am not really qualifying or have not really qualified what easy is. One of them is hard, and the third one is impossible. So one of them is easy, one of them is hard, and the third one is impossible, this is slightly more clearer, I think.

So, what do I mean by easy and what do I mean by hard, and what do I mean by impossible, and how are we able to infer about each of these things? How can we reach these conclusions that one of them is easy, one of them is not easy, one of them is hard, one of them is impossible. The meaning of these statements itself, I have not defined right now, but I am just saying that one of them is easy, one of them is hard, and one of them is impossible.

(Refer Slide Time: 07:39)

How can we make such conclusions? } GOAL

First, we need to understand what computation is. In this course, we will try to explore: **What are the fundamental capabilities and limitations of computers?**

Computers have evolved significantly throughout the years. But the theory of computation has remained applicable and relevant throughout.

Modern digital computers were developed around World War II. The theoretical

So, the goal of this course is exactly this. We want to be able to understand computational questions like these, and we want to be in a position to say that this is easy, this is hard, this is impossible by computer, etc. First of all, we have to define what is easy, what is hard, what is impossible, and then we should be able to systematically come up with a reasoning that will tell us, this is not easy, this is hard, this is easy, this is impossible, etc.

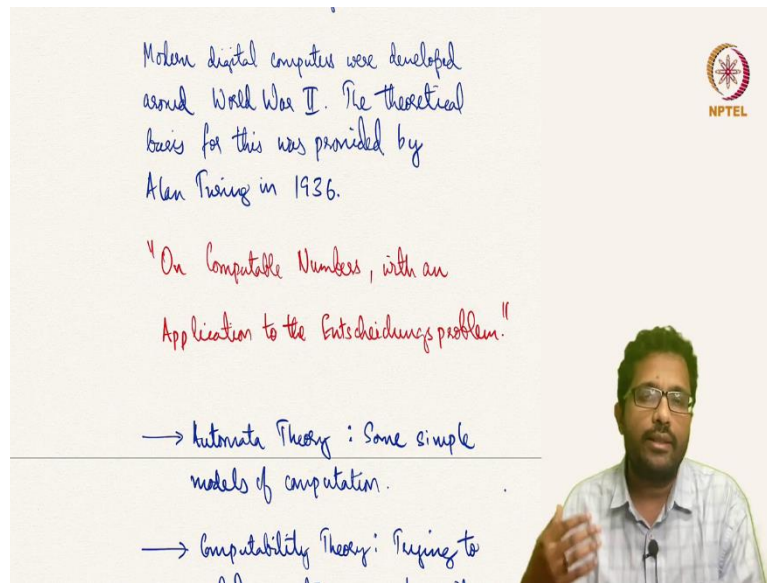
As a first step we should understand what is computation. This is also what we will see in this course. What can computers do? What is a computation? What can computers do? What can computers not do? We will try to see these things over the course. One thing that all of us know is that computers and electronic devices in general have technology that is so rapidly evolving.

Just to give a small example, the amount of RAM I have in my mobile phone now is more than the amount of hard disk space that we had in desktop machines, maybe let us say 20 years back or 25 years back. So the hard disk space in 1997, 25 years back, is now less than the RAM in the mobile phone these days. So, computers have evolved significantly but what we will see during this course is a theory of computation.

So, these are mathematical foundations of what is a computation. We will try to understand this. And this is fairly abstract, which is not really tied to a specific RAM size or memory size or the capacity of hard disk or whether the hard disk is solid state or magnetic. So this theory of computation has remained applicable throughout.

So, just because some speed has increased or memory has increased, does not mean that the theory becomes irrelevant or inapplicable. So that has remained constant throughout, and that is what we will see throughout during the course. Even though there is the field is rapidly evolving in terms of what computers can do, and how fast they can do, the fundamentals have not really changed.

(Refer Slide Time: 10:31)



Modern digital computers were developed around World War II. The theoretical basis for this was provided by Alan Turing in 1936.

"On Computable Numbers, with an Application to the Entscheidungsproblem."

→ Automata Theory: Some simple models of computation.

→ Computability Theory: Trying to

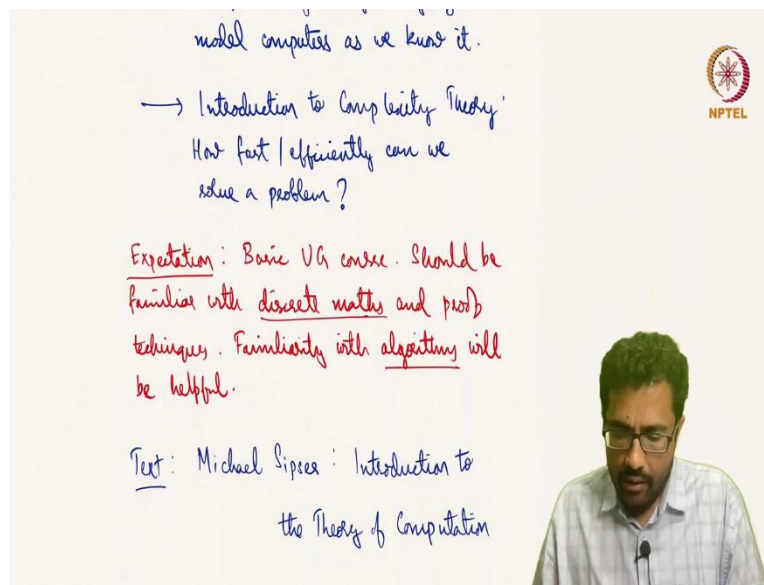
Just to give you a brief overview, digital computers as we know them or at least the current architecture or the current framework of digital computers evolved during the Second World War. One of the reasons for this to happen is that there was a lot of encrypted communication that went on during the Second World War.

So, the different sides used to send messages, secret messages which were encoded. So even if the opposition, the opponents could get hold of these messages by intercepting the radio channels or let us say if it was a physical message by intercepting the messenger, they had to actually decrypt these messages. It is not an easy thing, so this required lot of technology to be developed at that point, and so it is not really a coincidence that the computers evolved significantly during the World War II.

And Alan Turing himself was involved in a lot of code breaking in the second world war and he is one of the people who were responsible for providing the theoretical basis for computation. And a significant amount of what we will see during this course, will involve his contributions.

So, the models of machines, the models of computers that we will use in this course are called Turing machines, which were devised by Alan Turing. And the paper in which he actually proposed this model, came out in 1936. It is called “On Computable Numbers, with an Application to the Entscheidungs problem”. So, I will not get into what this problem was now, but this paper gave the theoretical basis of what computers can do.

(Refer Slide Time: 12:42)



So, what we will see during this course is, we will first see some simple models of computations which are not really equivalent to the modern-day computers. These are some simple, somewhat rudimentary models of computation which are not as powerful. These are called automata. We will see two types of automata, one is called finite automata and one is called pushdown automata, and we will see what these machines can do.

Then, we will move to computers or Turing machines, and we will see computability theory. So, we will try to model this. As I said already this will be the theoretical model of what the current modern computers can do. And we will try to understand what Turing machines can do. So this leads us to computability theory, so we will be able to say things like this problem is computable, and this problem is not computable. Computable meaning it can be solved using computers.

And finally, we will come to complexity theory, where we will only deal with how fast or efficiently can we solve a problem. Suppose we have a computer, and we are using it to compute a certain thing. Let us say we are trying to find the shortest path from home to office. Now, how fast? Maybe there are multiple ways of finding the shortest path. So what is

the fastest way? What is the way that we will use the least amount of space, and fastest meaning least amount of time.

We are trying to optimize on the resources. So two main resources that one considers during computation are space, which is how much memory we use, and time, how quickly we get the answer. So this will be dealt with in the complexity theory part. Complexity theory itself is a vast area, and towards the end of this course, we will try to briefly introduce this topic.

So, both of these topics are related, complexity theory as well as computability theory. And complexity theory also has a lot of applications in the sense that, for example I just mentioned cryptography already where we send encoded messages to the other party. This was used during the war, but it is also used every day now.

So, when you sign on to your email account, when you sign into your bank account, we are seeing something, the information in our email inbox or the information in our bank account details, it is displayed in our desktop or on our computer. But we do not want anybody in between, like even our internet provider or any of these in between servers to see what is going on, we do not want them to know our bank account balance.

So, these messages are encrypted. We want to encrypt fast, but we want the encryption to be strong, so we want it to be sufficiently hard. So, this requires some understanding of some problems being hard. So we want to know which problems are hard. Complexity theory will give us some way to analyse, the easiness or difficulty of problems. So, easiness, difficulty in terms of how much resources are required.

So, this is a brief overview of what we will see during the course. So, what is the expectation from this course? So, this is a basic undergraduate course. One thing that will be greatly helpful is if you are familiar with basic discrete maths, like sets, relations, correspondence, injection, bijection, etc.

Then proofs and proof techniques. So basic logic in trying to understand and reasoning logical arguments and proof techniques, some simple techniques like induction, etc. will be helpful. If you are learning algorithms or have learnt algorithms, the theory of algorithms that will also be helpful for the course.

(Refer Slide Time: 17:24)

be helpful.

Text: Michael Sipser: Introduction to the Theory of Computation

Before next lecture, read Chapter 0 from textbook: Sipser. Brush up basics like Sets, Functions, Relations, Graphs, Logic, Theorem, Proofs etc. Try out simple exercises.

The textbook that we will base this course on is a textbook by Michael Sipser, who is a professor at MIT. The name of the textbook is “Introduction to the Theory of Computation”. This is the name of the textbook that we will use during the course. Of course, in this NPTEL format, we will provide all the necessary documents and all the notes will be shared, of course. So, this is a basic outline of the course.

And, so before we move on to the next lecture, one small thing that will be helpful if all of you do, is to actually read chapter 0 from the textbook by Michael Sipser. It has a lot of basic information that you would have already seen the discrete maths course like sets, functions relations, theorems, proofs, etc. and fairly simple things, so maybe it will help if you go through these details and familiarize yourself.

These are just basic discrete math stuff that we expect you to be aware of, so that the rest of the course is comfortable. So please have a look at chapter 0, and maybe solve some, one or two simple problems. So, these are basics of discrete math. These do not really have anything to do with theory of computation but learning these will help us learn theory of computation easier and better. So, please go through these before we come to the next lecture.

And finally, I just want to get back to the three problems, and want to say which one is easy, which one is hard and which one is impossible? So, problem 3 happens to be the easy one, which you could have guessed, because Google Maps kind of does this. It tells us which is the shortest way from A to B, and this sits on everybody's phone or computers, and does this efficiently, so that is the easy question.

The hard question is problem two, the three colouring, It so turns out that this problem is hard. And the first problem is the impossible one. Again, the definition of easy, hard, and impossible, etc. we will see during the course. So, that is it. That is all that I have to say in this lecture, this brief introductory lecture. In the next lecture we will start with some basics that will help us progress for the course. Thank you.