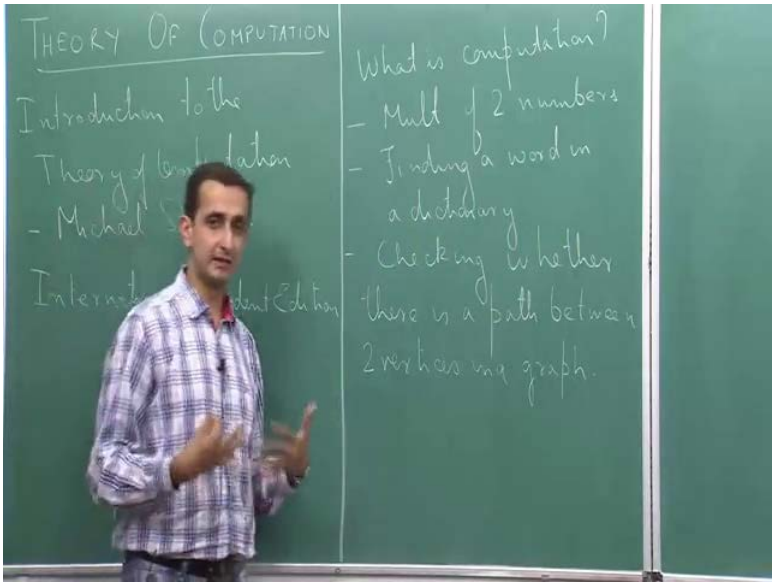
Theory of Computation
Prof. Raghunath Tewari
Department of Computer Science and Engineering
Indian Institute of Technology, Kanpur

Lecture – 01
Introduction to Finite Automata

Welcome everybody. This is a course on Theory of Computation. My name is Raghunath Tewari, and I will be your instructor for this course.

(Refer Slide Time: 00:32)



This is the first lecture, in this lecture we will give a high-level introduction to what theory of computation is, what kind of questions we are trying to answer in this course. And we will also look at one of the first basic models of computation that is a finite automata. And we will try to understand, what finite automaton is the one that, I will be following is the International Student Edition.

First let us understand, what we mean by computation? What is computation? Computation is the step-by-step solution to a problem. Suppose, we are given two numbers and we want to multiply the two numbers, how do we go about multiplying

them. So, there are known methods, there are known algorithms that we use to multiply two numbers in order to get their product. This is an example of computation.

Similarly, con sider that you have a dictionary and you want to search for a word in the dictionary. How do you search for that word? So you try, so in a dictionary the words are ordered in an alphabetical order. So, you just open random page and depending on, whether your word is before that page or after that page, you keep on looking at the dictionary, you keep on reducing your search base until and unless I mean until you find the word that you are looking for.

These are some of the example, so multiplication of 2 numbers, finding a word in a dictionary. Another example is, suppose you are given a graph and you are given two vertices - two fixed vertices in the graph, let say s and t. And you want to find out whether there is a path from s to t in the graph, so this is popularly known as the graph reachability problem. This is another example of a computational problem.

Checking whether there is a path between two vertices in a graph, so a standard algorithm to answer this question is you start a graph traversal algorithm. Let say, depth for search or breadth for search from the start vertex that is s and you keep on continuing, you keep on traversing the graph until and unless you hitty. If you hitty, you know there is a path from s to t; otherwise, you can say that there is no path from s to t in the graph.

The point is that for each of these problems, we can come up with a step-by-step method or what is known as an algorithm that helps us answer the question either in a positive manner, or in a negative manner. In the second case, we want when we are searching for a word, it actually allows us to come up with a solution, I mean whether so what is the word or what is the meaning of the word that we are looking for.

Now the next point that I want to discuss is when we are looking at computational problems, what model or where what kind of computational devices do we use to compute our solution or on what computational devices do we run our algorithm.

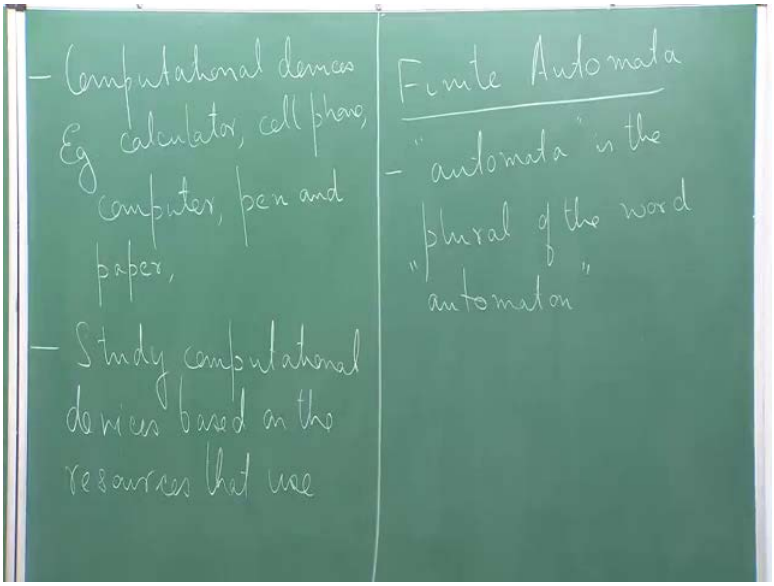
(Refer Slide Time: 05:20)

Again there is a host of computational devices that are there. For example, a calculator is one of it is the simple computational device that let us you perform basic arithmetical operations, sometimes even a little more. The modern cell phone or your smart phone also has the host of features using which you can compute in fact, a lot of things. In fact, the modern smart phone is almost equivalent to that of a computer.

Of course, a computer is a very popular example. And if you think about a little bit, for example, when I look at the first computational problem that is much you can of course,, multiply two numbers using a calculator or your cell phone or a computer, but you can also do it on pen and paper. So your standard pen and paper also is a computational device, because you can use that to solve a certain computational problems.

In this course, we will study computational devices, we will study computational devices based on the resources that they use. And for these computational devices, we will study their power and limitations; more specifically for a computational device, we will try to answer what kind of problems can be solved by these devices and what kind of problems cannot be solve. And for both these questions, we will give proper mathematical proofs in order to substantiate our answers.

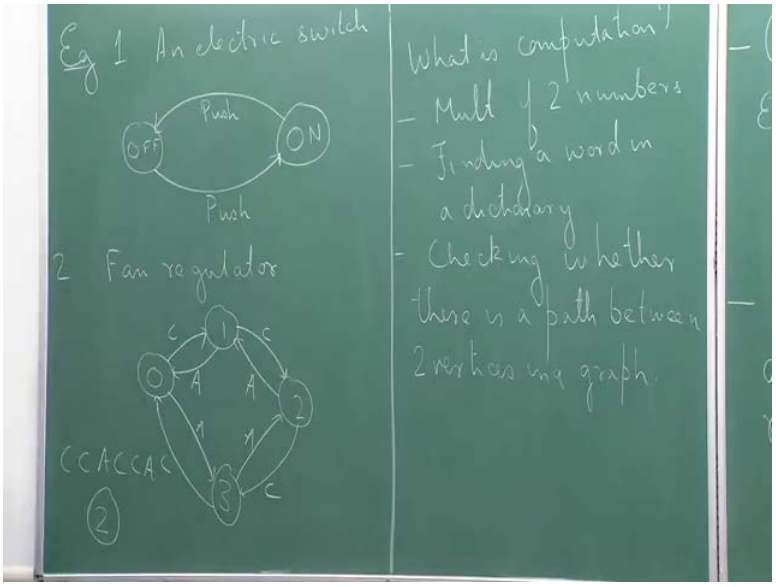
(Refer Slide Time: 08:40)



So, next we come to the first kind of computational devices that we will be looking at in this course which are known as finite automata. So, finite automata is a computational model which has in some sense a finite amount of memory, more specifically the finite automata has states and the number of states in such a model is a finite number, it does not depend on the size of the input that is given to it. Before going into what we mean what is the formal definition of a finite automata and how it works, I will give some informal examples.

In fact, I will give several informal examples in order to motivate what are the finite automata, how it works what is how much power such a model has and then we will come to the actual formal definition. Before that just a small note, so “automata” is the plural of the word “automaton”. So we have a single automaton and the plural form is automata.

(Refer Slide Time: 10:48)



Let us look at our first example of finite automata. An electric switch, so all of you have seen what an electric switch is, how many states does the electric switch have or at any given point of time, what can be the position of an electric switch. An electric switch is either in the on position or in the off position at any given point of time. So, these are what we call the states of the electric switch, so it can either be at the off state in which case whatever electrical device that is connected to it is not working or not getting electric supply, or it can be in the on state in which case the device is getting electric supply.

And how does the electric switch transit from one state to the other, so it happens per basically by the push operation that you perform with your finger. So if your electric switch is in the off state, and you push the switch it goes to the on state; and from the on state, again if you push the switch it goes to the off state. The point to be noted here is that so it gets there are two states that the switch has an off state and an on state. And the operation or the input that state can get or the value that we can give to our state any of the give the two states that are there is the push operation, and it basically flips between these two states. So, you can very easily and see that let say you start at the beginning, your switch is in the off state, and you give an odd number of push operations you will

end up at the on state; and if you given even number of push operations, you will be at the off state.

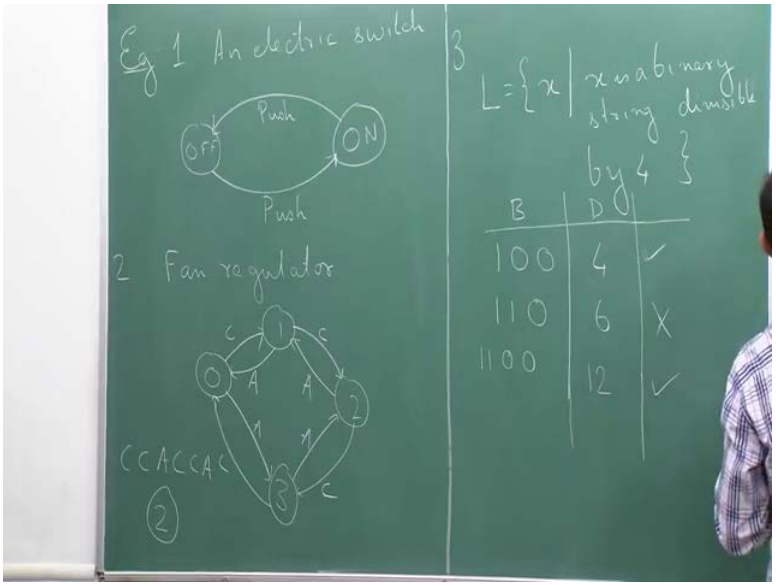
Similar examples, which is slightly bigger is that of let say fan regulator. So imagine you have a fan regulator, which has a off mode, which we will call the zero state, and it has three different modes low, medium and high. And basically it can transit from high again back to the off state. So, it is a circular fan regulator. So, we have four states; so we have 0, 1, 2 and we have 3. So, if I turn the regulator clockwise from 0, I denote that with c, it goes to state 1 or it goes to the low state.

From 1 again, if I turn it can clockwise, it will go to 2. From 2, if I turn it clockwise, it goes to 3. And from 3, if I turn it clockwise, it goes to 0. So, this is how it can go from one state to the state which is immediately higher to it. But note that for a fan regulator, we can also turn it the other way in the switch there is only one operation push operation, but in the case of fan regulator, we can go from 0 to 1 as well as we can go from 1 to 0, and the same for all the other state.

In a similar manner from 1, I can come to 0; if I go anticlockwise. From 2, I can come to 1, if I turn the regulator anticlockwise and the same for from 0 to 3. Here, basically we have two different operations that we can perform on each state, and we have a set of four states. So, for example, if I give you a sequence of the following operations let say CC AC CA which state will be landed.

So imagine, suppose if we start at state 0, and we perform these set of operations where will be end up. So after going clockwise, clockwise, anticlockwise, we will be at state 1; and then again we go clockwise, we are at state 2; we go clockwise, we are at state 3; we go anticlockwise, we are at state 2. So, we end up at state 2. Similarly, you take any other sequence of clockwise and anticlockwise operations; it will give you a unique state in which you will end up after you perform those operations.

(Refer Slide Time: 16:33)

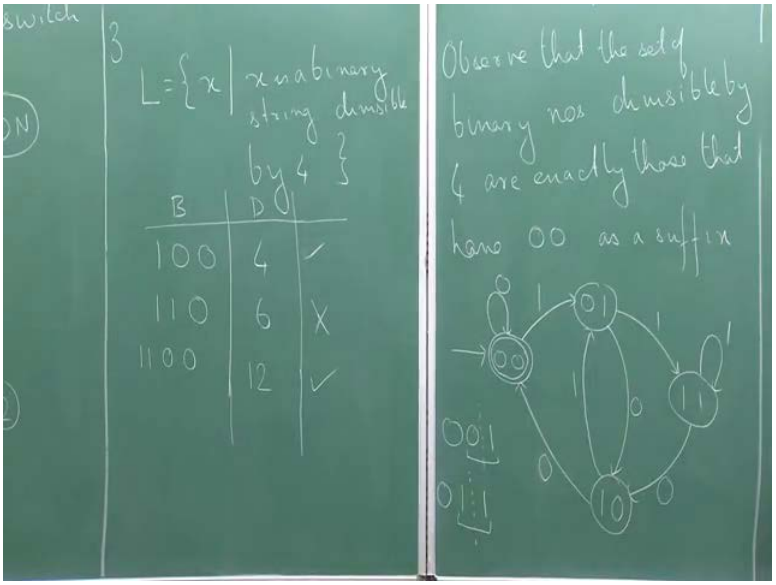


So, now let us look at a slightly more mathematical example. So, we look at the following set of strings. So, we will look at strings which are binary strings that is strings consisting of zeros and ones. And we look at the following set of strings, which we call L x is, it is set of strings x such that x is a binary string divisible by 4. So, let us look at some of the strings, let us look at some examples, so let us look at some strings which are in L and some strings which are may be not in L. So, if I take the string 1 0 0, so this is the decimal equivalent of this string is 4. So, here we have the binary and let say that here I write the decimal equivalent.

And here I will mark whether the binary strings belongs to the language L or not, or belongs to set L or not. So 4, of course, is divisible by 4, hence it belongs to the set L. What about 1 1 0? So, the decimal equivalent of 1 1 0 is 6; and 6 is not divisible by 4, hence it does not belong to the set L. If you look at 1 1 0 0, what is the corresponding decimal string, the corresponding decimal string if you just work this out, you will see that it is 12; and 12 is divisible by 4, hence it lies in the language and so on. So, if you just spend a little time, and if you just try to work it out, you will realize that the set of strings in the line or in this set L is exactly those strings which end with two zeros.

All numbers that are divisible by 4, all binary numbers that are divisible by 4 have two zeros in the end. And all numbers which are not divisible by 4 will not have two zeros in the end; it can end with either 0 1, 1 0, or 1 1 depending on what the number is, but not 0 0. So, this is the characterization that we will use in order to study this set.

(Refer Slide Time: 19:58)



So, observe that the set of binary numbers divisible by 4 are exactly those that have 0 0 as a suffix. So, in other words, we do not care what is there in the first if we have a n bit string, we do not care what is there in the first n minus 2 bits, they can be anything, but the last two bits must be 0 0. So, now, the question is can we design a finite automata that accepts all strings that are in this language. In other words, can I design a finite automata such that if I start at a given start state then if the number is divisible by 4, I will always end up at what we call an accept state.

And if the number is not divisible by 4, we will end up at a state which is not an accept state. So, before we even proceed to construct finite automata for this set, what do you thing the state should be or what kind of information should the states include. So, if you look at the first two examples, in the first example the states encoded the off and on condition of a switch, in the second example the fan regulator - the states encoded the different modes that regulator can be.

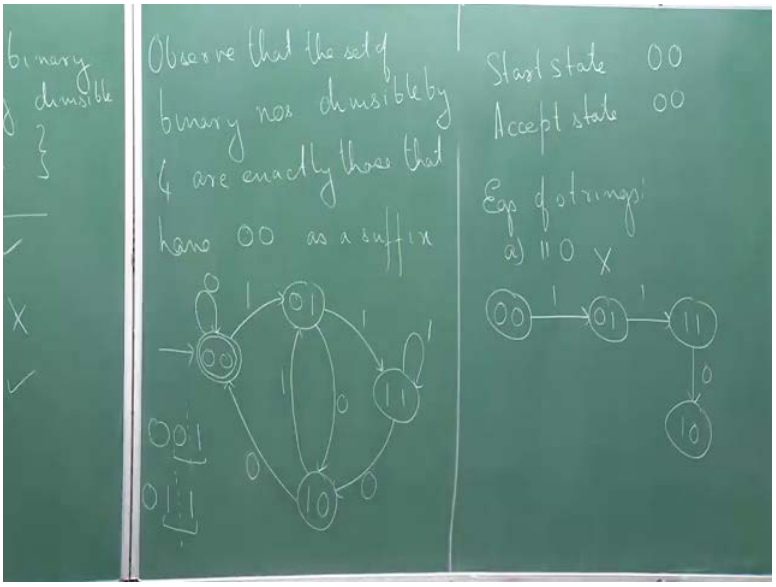
In this example if you look at it carefully the last two bits, so what are the different positions or what are the different values that the last two bits of a binary number can take. So, the last two bits of a binary number can either be 0 0, or it can be 0 1, or it can be 1 1, or it can be 1 0. So, these are the four different values that can be taken by the last two bits. So, now, if I am at a state 0 0, so I will surely say I have visited the number I am scanning the number from the left hand side, and I am at a position where the last two bits is 0 0. And the next bit that I see is a 1.

So where do we go, so we have 0 0 and we see a 1, so then we go to the state 0 1, because the last two bits was 0 0. Say I have a I have seen up to 0 0. And now I see one after this, so now, the last two bits that I am seeing are these two bits which is a 0 1. Similarly, from zero one let say that at a given point of time I have zero one has the last two bits that I have seen and now I see one after this where do we go to. So, now, the last two bits become 1 1, so we go from 0 1 to 1 1. And now similarly, you can see that from here from 1 1, if I see a 1 again 1 1, because it continues.

And from 1 1, if I see a 0, I will go to 1 0. From 1 0, if I see a 0, I will go to 0 0. And from 1 0, if you see a 1, we will go to 0 1. From 0 1, so what are the other possibilities remaining. So, let us go through each of the states. So from 0 0, if I see a 1, I go to 0 1. From 0 0, if you see a 0, we stay at 0 0. From 0 1, if you see a 1, we go to 1 1. And if you see a 0, we go to 1 0. From 1 1, if you see a 1, we stay at 1 1; and if you see a 0, we go to 1 0. And from 1 0, if you see a 0, we go to 0 0; and on a 1, we stay at 0 1. So these are all the possibilities that can happen.

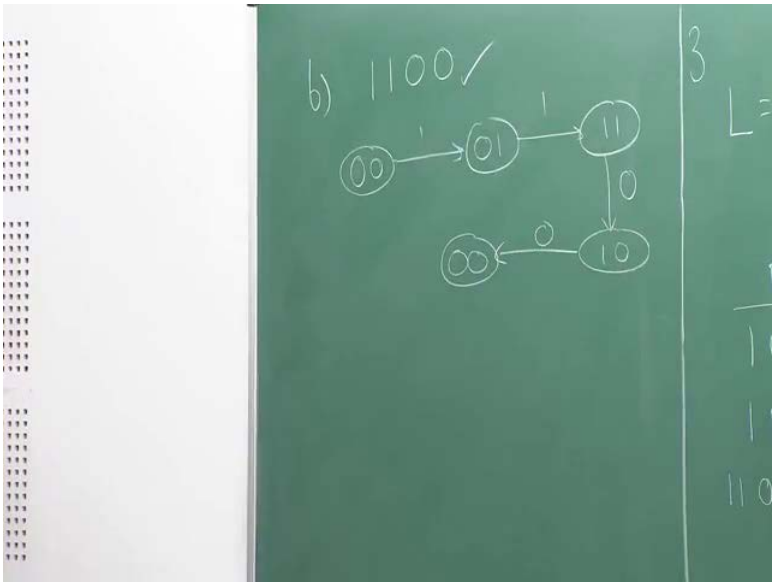
And, now a we start at the state 0 0, because this is when we have not read anything, so we can assume that the numbers so the string which it has consist of no symbol is divisible by 4. And depending on what we see from here on, we will take us to any of these four states. And finally, if we end up at the state 0 0 again, so I will mark it with two cycles we say that the number is divisible by 4. And if you end up at any of the other three states we say that the number is not divisible by 4.

(Refer Slide Time: 26:48)



Let us just work this out so our start state is in this case is 0 0; our accept state in this case is again the state 0 0. And let us look a couple of examples. So, suppose if you get the string 1 1 0, so on the string 1 1 0, so initially we start at the state 0 0; if I see a 1 by this automata, I go to the state 0 1. On the next 1, I go the state 1 1; and on the next 0, we go to the state 1 0. Hence the string 1 1 0 is not accepted by the automata, because we do not end up at the accept state.

(Refer Slide Time: 28:22)



Another example, let us look at, so let me go to this side. So let us look at another string. So, let us say we have a string 1 1 0 0. So from 0 0 on 1, we go to 0 1; from 0 1 on a 1, we go to 1 1; from 1 1 on a 0, we go 1 0. And if you look at the automata from 1 0 on a 0, we go to 0 0. Hence the string 1 1 0 0 is acceptable. So, I will stop here.