

Digital And The Everyday: From Codes To Cloud
Prof. Shrisha Rao
Department of Multidisciplinary
International Institute of Information Technology, Bangalore

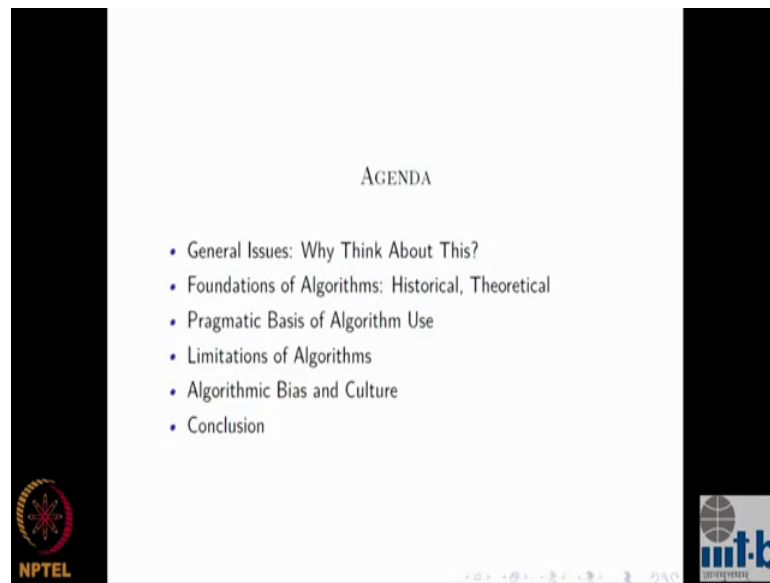
Lecture - 03
Socio-algorithmic processes & the Everyday-Part 01

Good morning once again and thanks to Bidisha, and my colleagues for inviting me to give this talk. This is a slight variation on a talk that I have given a few times before including on this campus, so especially some of the local yokels who probably heard me say this and in fact I tend to spout off about this at all occasions so you probably have heard me say that also. And one thing is this is not a polished subject. Now, I teach certain classes here of course in my role as a member of the faculty and then there are textbooks, there are lesson plans, there are syllabi, there are established conclusions fixed theories, well solved and well posed problems with proper answers, this is not one of those.

This is in fact in some senses a very incomplete or superficial talk, but at the same time, I think I am trying to ask some questions, which I want all of us to think about. We may not have the answers to those questions yet, but these are in my view very important questions, and this is something that we should be thinking about as we go along. And it is very good that I see a lot of people here who are not just IT professionals, but also people who have other kinds of relevant background and that is something that I think will help us all.

So, and one other point of note I do not know how the NPTEL folks will go with this, but I am in general very interruptable. So, if you have a question, you can stop me right there and ask and you do not need to wait till the end. So, anyway to start with this, this is the talk. And hi once again, my name is Shrisha Rao. And this is once again a topic which I think is quite important, but it is not a finished polished subject where there can be a complete set of fixed lessons to be learned. It is something where there are very important questions to be asked and for some of those questions we have answers for a lot of them we do not and that is exactly the point that is why we need to think about these things.

(Refer Slide Time: 02:17)



AGENDA

- General Issues: Why Think About This?
- Foundations of Algorithms: Historical, Theoretical
- Pragmatic Basis of Algorithm Use
- Limitations of Algorithms
- Algorithmic Bias and Culture
- Conclusion

The slide features the NPTEL logo in the bottom left corner and the IIT-B logo in the bottom right corner. A navigation bar with icons is visible at the bottom center.

So, one of the general issues is why do we even bother with this, what are the social aspects of algorithms, and why are they important to us, and then what are the foundations of algorithms historical and so on. Now, this will probably be a repeat to some of you who have studied computer science, but it may be somewhat new to some of you. And then what is the pragmatic basis of algorithm use, there may be a certain amount of theory, but we also have to think about how they actually work in practice. And then are there any limitations, yes, it turns out they are there are. And then what about algorithmic bias in culture something that my good friend and colleague Bidisha just alluded to and then some concluding remark. So, this is the rough agenda to start with.

(Refer Slide Time: 02:57)

2/36

WHY ALGORITHMS MATTER IN SOCIETY

- Computational systems are everywhere in modern society, and algorithms drive them.
- Perhaps on account of *scientism* in common thinking, algorithmic processes, and the systems that enable them, are seen as infallible and beyond reproach. (This is particularly evident in government and corporate policies that seem to promote the use of computing systems as a cure for all ills.)
- Algorithms are, in some senses, *plus royaliste que le roi* (more royal than the king)—not only are they driving society in some ways, but individuals and society at large are now expected to cede primacy to them. Often, we can no longer think or function independently, which should bother us.

NPTEL

mit-b

mit-b

So, the first thing is why do we have to think about this. And one point to this is that computational systems are everywhere. Thanks to newer technology is the fact that we are increasingly automating a lot of things that did not use to get automated before, there are hardly any pure technical systems anymore that do not use some kind of IT in them right because of iot and many other technologies we do not have pure engineering systems anymore almost nowhere. And even where there are they are soon going out and we will have some type of networking coming into them. So, computational systems are everywhere; and algorithms are used to drive them. And one other important point to note is that what is the big basis what is the need for AI to become so big all of a sudden.

AI has been there as a theoretical discipline for a very long time since the 1950s, but why is it so big all of a sudden. Have you ever thought of that? One of the big reasons for that is that so many people require services, so many people require IT systems, so many people require IT enabled services. Now that there are not enough people to support all those services physically or you cannot really deploy people like telecom center, call center kind of people and deploy enough support staff to deliver all the services you need to all the people who want to use them that is the big need for there to be AI and so on.

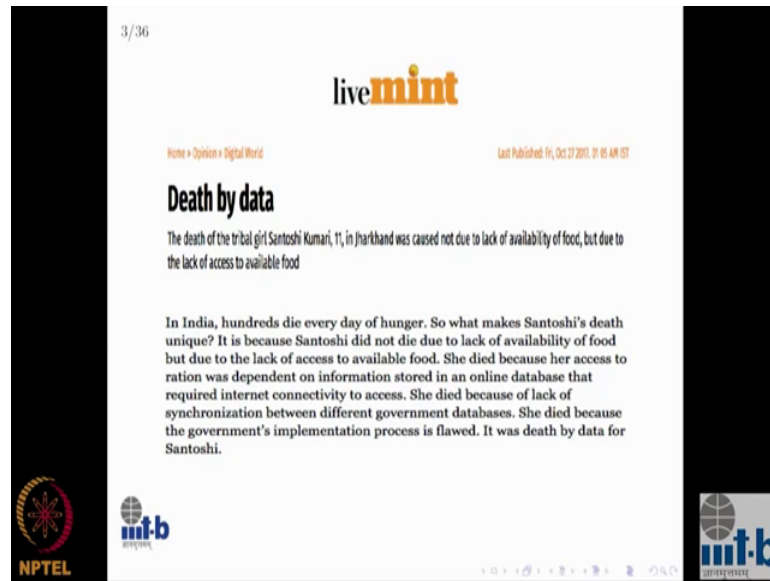
And one other important issue is there is this term called scientism which really refers to I am a scientist, I think many of us are scientists, we all have a very high regard for science, and I am not at all suggesting otherwise. But scientism we must admit has its

limitations. And that refers to the way of thinking where you believe that science or scientific systems are infallible where you are willing to take a blind risk, where you are willing to believe good about something, simply because it is supposed to be scientific in its basis. And that I think is problematic. And I think we all have to agree it is problematic and that is also because of government policies and corporate policies, where science or technologies are advertised as being the cure for all ills and that is a big problem. Unfortunately that is happening in our country it is also happening worldwide and that is actually a case where algorithmic processes do not necessarily help society in the way that they should.

And that in all means that you have this French expression called *plus royaliste que le roi* which means more royal than the king. And the English equivalent of this would be more catholic than the pope. In all senses what it means is that in some way computing systems, algorithms have supplanted human thinking. We are in the process of losing our human dignity and our human nature even. My human judgment is now eclipsed if not supplanted by algorithms. And in some sense that should bother all of us, it affects human rights, it affects our individual sense of who we are as people, it can affect the quality of life and the quality of our society.

In general, whenever we have had loss of freedoms we have been worried about it. As a society throughout history when people lost their freedoms when they were told that they no longer had the freedom to do something that they otherwise should have done, they minded it and so should we that is the point here. So, we should think about that.

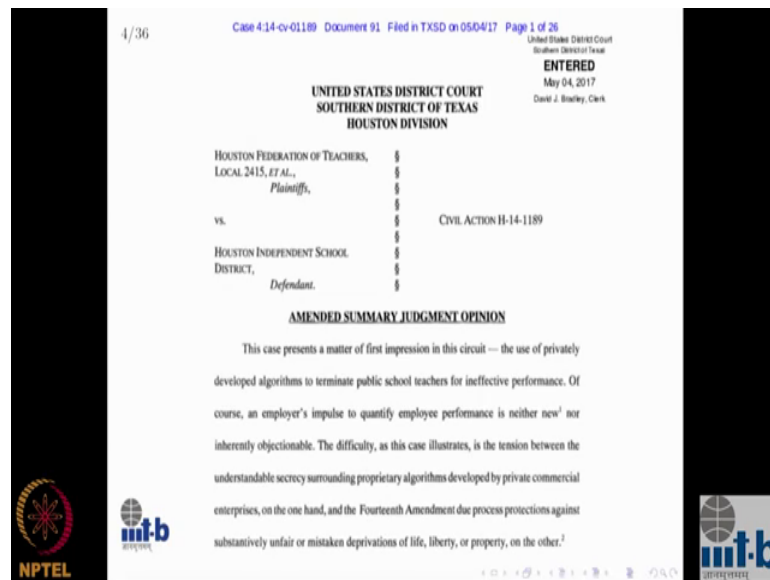
(Refer Slide Time: 06:25).



And couple of actual examples of this. So, this is an actual news article that came out a couple of months ago, a few months ago, where this actually happened in our country. So, this is an example of a child who was actually who actually died of starvation, and of course, that is a great tragedy. What was worse was that this tragedy did not have to happen. Most tragedies do not have to happen, but this one especially did not have to happen because it was because of IT. So, there was a mismatch in some government databases and because of all this aadhar mandate, and Digital India and so on, some ration that the family had was supposed to get, could not be delivered, she did not get and that poor child died.

Now, one of the lessons, I learned maybe not that well back in the day when I was a student was IT is not pure algorithm is not algorithms are not pure they are not simply a theoretical abstraction that just sit in some theoreticians mind. And do not really it is not really abstract philosophy in that sense they have real consequences. There are people who can be helped or badly hurt by them, and this is one concrete example of that. And of course, there are other examples which are possibly suitable in a different context, but this one is especially startling for us because it happened in our country in a way that we probably do not want to see ever happen again. So, this is one.

(Refer Slide Time: 07:47)



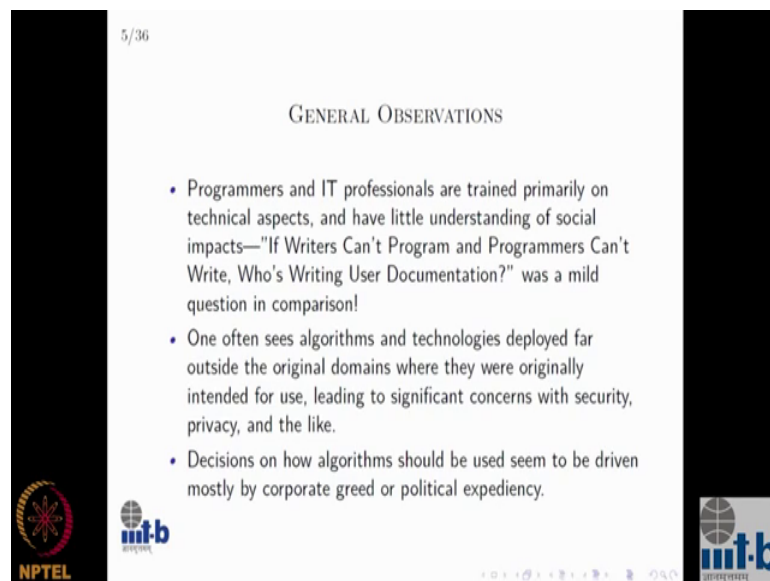
And a second example of this. This is actually from the US also from this year, where in the Houston school district, the public school teachers were being judged or they are still being judged by some computing system. Their quality of performance, how well they do as teachers is now being evaluated by that school district using some proprietary algorithm and some proprietary computing system that no one understands. There is some software, there is some vendor who has supplied that software, that software is actually used to make evaluations, and that those evaluations actually determine which teachers get promoted which teachers get fired etcetera, etcetera. There are actual career consequences to these people based on this.

So, what happened was the school district was sued and this is an actual judgment copy from that judgment. There was a summary judgment this is not quite settled, but this was this actually the one level of judgment did happen in May. This actually was I think filed in 2015 or 16, and it was litigated late 2016 and the judgment was delivered this May. So, the school district was sued by the teachers in that school district saying look you cannot do this. If you are actually going to judge us, if you are going to say that we are not good enough for we do not need to be employed here anymore, then you need to tell us how you are judging us. We need to know what those algorithms are, what the metrics are, how they are arrived at, we need to understand if they are fair. So, this is what this is.

And it is actually something that is probably going to happen more and more often and there is some interesting language in this that you can look up also. And I have given a reference to this and some other things at the end, I will mention that so anyway. So, just for your curiosity this was essentially set aside by the judge and said no this and he gave the summary judgment where he denied the plaintiffs claims. And he essentially set it aside, but it is not quite done yet those teachers will probably go on and file an appeal and it will go all the way to the supreme court at which point we can say there will be some case law on this topic, we are not quite there yet. So, this is still going on, this is not, the story is not yet over, anyway.

So, these are two examples, an example from our country where even very poor people can be badly hurt; this is an example from a more mature society where peoples careers and their performance can be judged by algorithms which we do not fully understand which are not revealed to us right. So, there are many such examples I just chose two because I did not want to go on and on just on this topic.

(Refer Slide Time: 10:19)



5/36

GENERAL OBSERVATIONS

- Programmers and IT professionals are trained primarily on technical aspects, and have little understanding of social impacts—"If Writers Can't Program and Programmers Can't Write, Who's Writing User Documentation?" was a mild question in comparison!
- One often sees algorithms and technologies deployed far outside the original domains where they were originally intended for use, leading to significant concerns with security, privacy, and the like.
- Decisions on how algorithms should be used seem to be driven mostly by corporate greed or political expediency.

NPTEL

MIT-B

MIT-B

And then there are some general aspects to this that we have to keep in mind. Those of us who have been trained in computer science such as myself, we are trained primarily on technical aspects. We do not really understand the social aspects of what we do. We do not really understand the social aspects of or the consequences of our work. So, there is a

very well known title from the 1980s saying if writers can program and programmers cannot write, who is writing user documentation.

And in fact, if you notice IT systems these days do not have any user documentation, your iPhone or whatever phone you use did not come with the manual did it, and that is because essentially the IT industry has given up on user documentation. This turns out to be a problem they really cannot solve. And unfortunately it is not just with user documentation that the problem arises. You have all these once again like Bidisha was saying this socio technical confluences where people who are trained in one or the other do not really know how to establish the whole thing, and do not really know how to explain the whole thing and that is one point of it.

The second part is also that you often see algorithms and technologies which are well outside the realm where were only originally intended for use. And one example of this would be with I think smart cars or self-driving cars. There are many examples like that that is just one of them. Now a self driving car or in fact, it does not even have to be self-driving take any modern high end automobile made by Mercedes Benz or BMW or your favorite maker whoever that might be that often has a car area network CAN, which is nothing but a LAN. Which is of course, something that we all know in a different context very well, but the assumptions that went into designing the LAN in an office setting or in a corporate setting or in an academic setting are very different from what applies inside an automobile. But because it is essentially the same technology there are certain concerns with privacy, you are not typically concerned about someone inside the LAN hacking the LAN, but you are worried in the case of a car. So, someone who has access to the key one time may or may not have access to the whole system or may not be somebody you trust with the whole system all the time things like that.

So, there are a lot of such things where algorithms and technologies are conveniently ported to a different domain without fully understanding the context of the domain, and how that actually works. And then how algorithms should use that is not a question that people answer based on social norms. They do not really think about what the effect on society will be, they are driven by corporate greed. The chip companies want to advertise how quick the chips are, some phone company wants to show you how good their phone is. And in all these cases they are not really thinking about the quality of your life, they are thinking about their bottom line and their profits which is ok, I mean we are all high

sense I suppose in some sense profit driven and we are all self motivated in that way, we are all selfish in that way. But nonetheless that does matter because in many cases you have evolution in IT systems which is not quite the way we think it should be.

(Refer Slide Time: 13:21)

1/36

ALGORITHMS: HISTORICAL BASIS

- The first recognizable algorithm is Euclid's algorithm for finding the GCD of two natural numbers.
- The name itself comes from Muḥammad ibn Mūsā al-Khwārizmī, a 9th-century Persian scholar and mathematician.
- His book *Al-kitāb al-mukhtaṣar fī ḥisāb al-jabr wa'l-muqābala* also introduced the term *al-jabr*, which eventually became *algebra*.
- Originally, Latin works would give procedures for number manipulation, solving quadratic equations, etc., with the imprimatur *Dixit Algorizmi*—"Thus spake Al-Khwārizmī."

NPTEL

mit.b

mit.b

So, some historical notes about this the first recognizable algorithm for anything was Euclid's algorithm for the GCD of two numbers, which we learned I think many years ago. For some of you I suppose youngsters it was much more recently. So, this is the historical first algorithm that anyone can identify. And this comes from the Gentleman's Name Al-Khwarizmi Muhammad ibn Musa, Al-Khwarizmi who was a 9th century Persian scholar and mathematician. He also wrote a book which gave rise to the name algebra also which introduced a word called algebra. And then this al-jabr became the word algebra.

So, essentially this gentleman Al-Khwarizmi his name gave rise to the word algorithm, and his book title gave rise to the word algebra. Now that has got to be an influential book. And in fact, Latin works for a very long time would quote this guy as an authority they would say Al-Khwarizmi said so. So, for a very long time this was a very influential mathematician and he had a very big influence even in Europe in the middle ages because of the quality of his work so anyway. So, this is the historical basis of algorithms this is where it all came from.

(Refer Slide Time: 14:30)

2/36

MATHEMATICS AND ALGORITHMS

- Mathematical reasoning is in part algorithmic, in that *constructive* methods in math are inherently procedural.
- Historically, some mathematicians have been hostile to *existence* proofs, as illustrated by Paul Gordan's critique of the young David Hilbert's work: *Das ist nicht Mathematik. Das ist Theologie.*
- It is also seen in Kronecker's remark, which translates as "God made the integers, all else is the work of man."

NPTEL

mit-b

mit-b

And mathematics is in large part algorithmic, in the sense that constructive methods in all in mathematics are inherently procedural like for example, bisect an angle in geometry they teach us various things where you do certain things and those are inherently procedural, procedural is nothing but algorithmic. And that was the tradition in algorithm or in mathematics for a very long time until about the early twentieth century that was the tradition in mathematics where everything was procedural, you had a fixed process to find something.

And there is a very famous critique by a mathematician called Paul Gordon of a young David Hilbert, Hilbert himself was a very famous mathematician in the late 19th and early 20th centuries. And when Hilbert came up with an existence proof, where he did not have an actual process to construct that object, he did not have a process to find that value, he just showed that such a value existed. So, Paul Gordon criticized this saying this is not mathematics, this is theology. In theology, they argue various abstract things, they do not actually have a way to demonstrate that such an object exists like reasoning about God, so that was what the critique was about Hilbert's work.

And there was also a similar statement made by Kronecker this was actually made in German the translation of that is God made the integers, all else work of man. In fact, now this would be made fun of it and in fact it is if you see this in a mathematics book today it will probably be in a pejorative sense will they actually make fun of Kronecker.

But the at the time what Kronecker meant was natural numbers and the process is for manipulating them those are proper, existence proofs and stuff that you actually just reason about in the abstract that is not good mathematics, that is not something that we actually can accept.

(Refer Slide Time: 16:15)




3/36

MATHEMATICS IS NOT ALWAYS CONSTRUCTIVE

Littlewood's perspicacity, Skewes' Number: $10^{10^{34}}$

x	$\pi(x)$	$\pi(x) - \text{li}(x)$
10^8	5761455	-753
10^{10}	455052511	-3103
10^{12}	37607912018	-38262
10^{14}	3204941750802	-314889
10^{16}	279238341033925	-3214631
10^{18}	24739954287740860	-21949554
10^{20}	2220819602560918840	-222744643
10^{22}	201467286689315906290	-1932355207

Table: Comparing Primes less than x , and $\pi(x) - \text{li}(x)$

But however, once again since the early twentieth century mathematics is not only procedural, and this is one once again a classic example of this. I am really not competent to explain this in great detail, but I will just go over this very briefly with you. Now, Littlewoods this guy Littlewood was a friend and colleague of G. H Hardy, Hardy being of course, the Cambridge mathematician who also worked with Ramanujan that is the name I think you know. And there are two functions we would not go into what they are π of x and li of x . And as x increases x being a natural number, π of x is this value and this difference also seems to keep increasing, so that li of x is actually larger and larger than π of x . So, the natural conjecture was even when you have larger and larger values, this keeps increasing right, so that was the conjecture that π of x is always smaller than li of x .

Now, this seems very natural given that even as you get to very large values this value keeps on increasing in this way right, but Littlewood had this great insight which is really uncommon saying that no, this is not always the case and in fact these two values keeps switching. They keep switching where one is larger than the other for a while then

the other one is larger than this for a while and so on and this switch happens infinitely often that was a fantastic result. Now, the problem is we get to such large values and we have not seen a switch and in fact this is actually getting bigger and bigger. So, when does the first switch happen and that happens at a number called the skew number which is ten to the ten to the ten to the thirty four this is obviously, much bigger than the list of any number of things you can have in the universe the protons neutrons and so on.

So, this is the time of the first switch. If you get this far then there will be a switch where $\log x$ suddenly becomes smaller than π of x . And then after that there will be an infinite number of switches beyond that. Now, this is not something you can establish constructively and in fact to this date we do not know as far as I know the first value at which the switch actually occurs this is actually the bound skew number. So, this is an example of an existence proof where in mathematics we know that something happens, but we do not know the precise value at which it happens right.

So, there is no algorithm for doing this, but Littlewood came up with this result which said it does happen I do not know where it happens, but it happens right. So, mathematics has moved away from the algorithmic aspect not completely, but mathematicians today do not really consider themselves as doing the kind of thing that Paul Gordon would approve of. So, there is a lot more of this kind of stuff happening in modern mathematics.

(Refer Slide Time: 18:57)

4/36

THEORETICAL FOUNDATIONS OF ALGORITHMS

- A *Turing Machine* is an abstraction of a computing device that manipulates symbols according to some rules, and has input, output, and storage.
- There are different models of computation (besides Turing machines), but all of them so far are found to be equivalent, and the Church-Turing thesis asserts that "effective computability" in the intuitive sense is the same as Turing computability.
- With this, an algorithm is assumed, in theoretical computer science, to be a procedure that can be executed by a Turing machine.
- Every algorithm is, in essence, a realization of a *computable function* from \mathbb{N} to \mathbb{N} .

NPTEL IIT Bombay

So, theoretical foundations once again those of you who are CS people do not need this too much, but I will just go over this briefly. A Turing machine once again which is an eponymous machine named after Alan Turing it is a model of a computing machine where it manipulates certain symbols or according to some rules it has the basic properties of computer input output storage. And there are different models of computation beside Turing machines, but all of them so far are equivalent to Turing machines no one has come up with a model that is different from a Turing machines they all have the same expressive power, they are all quote unquote Turing equivalent.

And the Church-Turing thesis says that effective computability is the same as being computable on a Turing machine right. So, in general, in theoretical computer science, which is a domain that I came from or that I do come from, an algorithm is exactly what you can do on a Turing machine. This is important to note because we have all these fantastic claims or counterclaims or whatever about AI or this or that, but as far as theoretical CS is concerned what does computability mean it means exactly what it Turing machine can do nothing more nothing less. And every algorithm is therefore, is essentially a function from the set of natural numbers to the set of natural numbers, it is a computable function from n to n that is what this is.

(Refer Slide Time: 20:12).

5/36

WHAT ALGORITHMS CANNOT DO

- Computing over the real numbers. (Turing machines only know natural numbers.)
- Logical operators such as *however* and *nevertheless*.
- The set of all functions from \mathbb{N} to \mathbb{N} is a strictly larger infinity than the set of all computable functions, so there exist uncomputable functions.

NPTEL IIT Bombay IIT Bombay

So, therefore, there are certain things algorithms cannot do. One easy example is computing over the real numbers, because computable functions are only from natural

numbers, natural numbers. And similarly there are certain logical operators that you and I use in our daily lives like, however and nevertheless there is no algorithm that can actually capture them. Similarly, the set of all functions from \mathbb{N} to \mathbb{N} is a strictly larger infinity than the set of all computable functions that is a very easy result to show, if you are interested. So, there are uncomputable functions we can actually come up with examples of them. Therefore, in the abstract even in the abstract realm, we know that algorithms do not actually capture everything. There are things which we cannot do using algorithms.

(Refer Slide Time: 20:55)

6/36

COMPUTERS AND MINDS

- "Strong AI" is the view that human level cognition is algorithmically possible, and in fact that the human mind is algorithmic in nature.
- The "mind-brain" identity thesis is philosophically controversial, and it is anyone's guess whether extreme predictions such as the Singularity will be realized.
- The computational theory of mind has some important ethical implications.

NPTEL

MIT-B

MIT-B

And to move onto more pragmatic aspects strong AI is the view that human level cognition is algorithmically possible. So, there are some scholars who claim this Daniel C, (Refer Time: 21:05), and so on. They try to give a physical interpretation of the mind, they actually say or think they propose that human minds human brains are nothing but Turing machines maybe with a very different physical architecture, but that is what they are. And anything that a human being can do a machine can hypothetically do. So, there is nothing really special about you as a person yes.

Student: Sir could you explain how algorithm works functionally as then (Refer Time: 21:31) output I mean I am not being able to poses that so far.

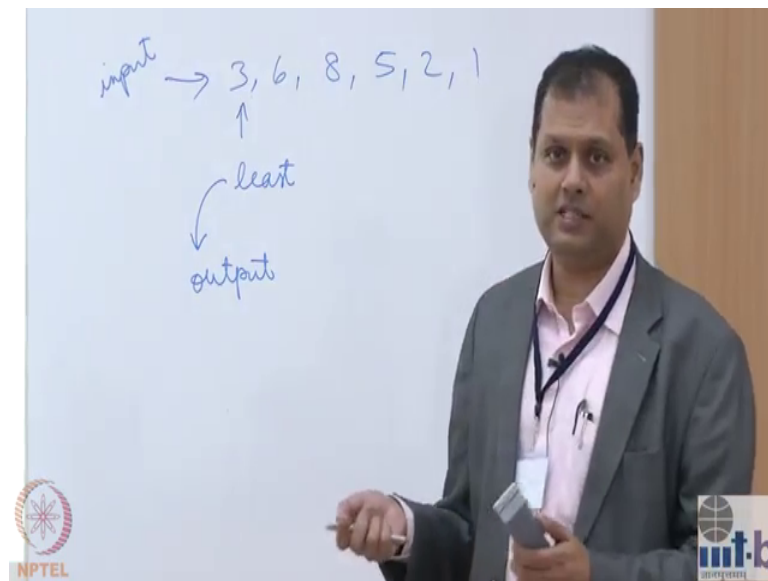
I may be missing your question to some extent, but can you clarify exactly what the.

Student: how structure algorithm I mean I am not an engineer I mean so if you could explain that.

Ok

Student: you also made a difference between pure computing and algorithm so at the very beginning, so if you clarify these two things.

(Refer Slide Time: 22:12)



Suppose, I have let us take a very small example of something I need to do. So, for example, if I have a list of numbers right, so if I have 3, 6, 8. I need to find the least of these values that is a classical; it is not even sorting, just to find the least value you know in an array. How would you do that? So, the way that classically algorithms proceed or the way that computer scientists have for the last seventy years process this is they think about ok if I am a human being and I do this. How do I do it, can I come up with a step by step process?

Now, once again the one classic example of this is we all know how to make a peanut butter and jelly sandwich right. Can you describe how to do it to someone who has never done it before? And more precisely can you describe how to do it to a robot? And it turns out that yes you can, but it is a very intricate process because a lot of implicit knowledge that I have has to now be put forward it has to be really described in great detail. Now, of

course, there are certain programming language concepts and so on that I am not going to get into data structures and so on.

So, basically once again if I need to find the least value what do I do, I say I will set this to be the least value, I will have a value that I say call the least, and then I will go step by step. And then if I see the next value as being less than this I switch. So, I said that to be the least, and I keep going all the way to the end. And whatever I am left with as the value in least is the least value in that array right, that is an example of an algorithm where this is your input, and then this at the end should be the output. Once again missing a lot of important details right.

So, the end if you look at or if you have I do not know how common this is in India, but you have Ikea furniture some assembly required, where you have furniture that comes to you not this way, but it comes to you with parts that you need to assemble. And then there are some instructions on how to assemble them, you need to put this screw there, and you need to do this and use this tool and that and so on that is also an algorithm in a very different context. Essentially, a well developed well-described process for doing something where you start with an input and get a well defined output that is an algorithm, right anyway.

So, coming back to all this. The strong AI is the view that human level cognition is algorithmically possible this is not really a scientific fact it is a theory, it is a kind of school of philosophy if you want to call it that and that it is also view that the human mind is algorithmic in nature. So, anything you call as being your own thoughts, your own emotions, your way of thinking, your way of processing something all that is ultimately algorithmic.

And this unfortunately is somewhat philosophically controversial, the mind brain identity thesis is not completely free of controversy, there are people who agree with it, some people who disagree with it and so on. And then there is a very well known book and a theory called Singularity by Ray Kurzweil where this gentleman proposes that you ultimately can get to a point where you can your mind can actually be downloaded into a machine and so on, and then you can have eternal life in that way in some sense. You are no longer the physical embodiment in human body you are actually going to live inside a machine as a program.

And then once we are not going to be discussing all that today by the way, just mentioning this because it is a relevant fact in the context of algorithms, it is not the point of this discussion. The computational theory of mind has some ethical implications for example, human rights, how do they change, if you consider that human being is nothing but Turing machine that is something that we can also discuss which we will not.

(Refer Slide Time: 26:08)

7/36

THE PRAGMATIC BASIS OF ALGORITHMS

- The stored program concept of Von Neumann: use a general purpose machine with a stored program, rather than a special purpose machine for each task. (Programs as well as data can be stored.)
- The general purpose machine is the *hardware*, and the stored program is the *software*.
- Algorithms of some kind are instantiated by the stored program.

NPTEL

MIT-B

MIT-B

In terms of the pragmatic basis of algorithms, this is something that we do think about. There is a stored program concept, which came about in the 1940s give or take Von-Neumann very well known mathematician made the observation that at the time there were computers being used for various special purpose tasks. So, you would have different machines physically different machines doing different things in different roles. And it was obviously, getting too expensive to replicate machines for each task that people wanted to do. And for von-Neumann inside was the stored program concept where you have a general purpose machine and then you give it the specific instruction you want depending on whatever task you want to do it, want it to do.

So that was the store program concept where instead of storing only the program or only the data for the program, you also could store the program itself. So, you do not have storage just for data, you have programs also being stored. And now of course, we are very familiar with this concept where the general purpose machine is the so called hardware and the stored program is a software. And depending on the software that you

run the same machine can do different things, you do not need one laptop to check email, a different one to run Firefox, the third one to do something else and so on, you have the same machine that can do multiple things.