IIT Madras

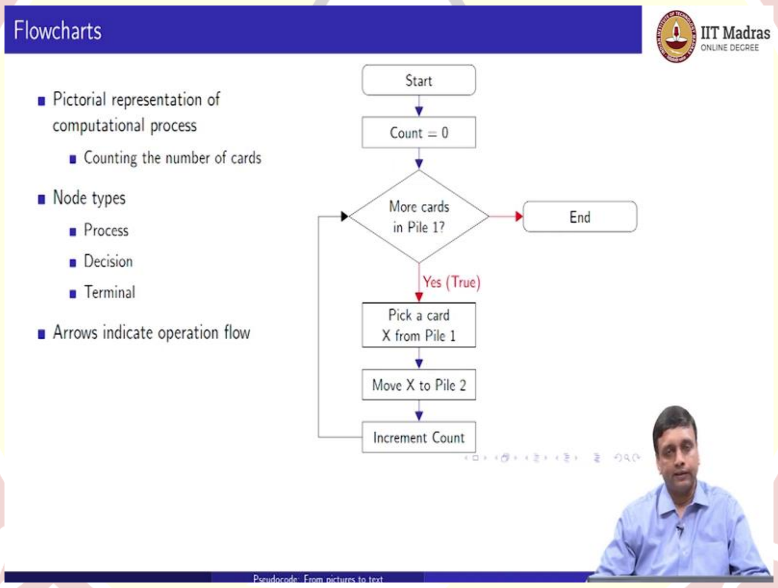
ONLINE DEGREE

Computational Thinking
Prof. Madhavan Mukund
Prof. G. Venkatesh
Department of Computer Science
Chennai Mathematical Institute
Indian Institute of Technology, Madras

Lecture – 2.9
Introduction to pseudocodes

So, we have seen that we can use flow charts as a way to describe algorithms precisely. Now, we will look at a different notation which we call pseudo code which is actually a textual representation of procedures.

(Refer Slide Time: 00:27)



Flowcharts

- Pictorial representation of computational process
  - Counting the number of cards
- Node types
  - Process
  - Decision
  - Terminal
- Arrows indicate operation flow

So, if you remember, we had drawn some flow charts for the algorithms that we had discussed in the class. The first one was the simple one to count the number of cards in a stack of cards.

So, here on the right, we have a flowchart which basically starts then sets a variable count to 0 then as long as there is a card to check, it picks it up, increments the count and moves the card to the second pile. And once all the cards have been moved to the second pile, then we stop.

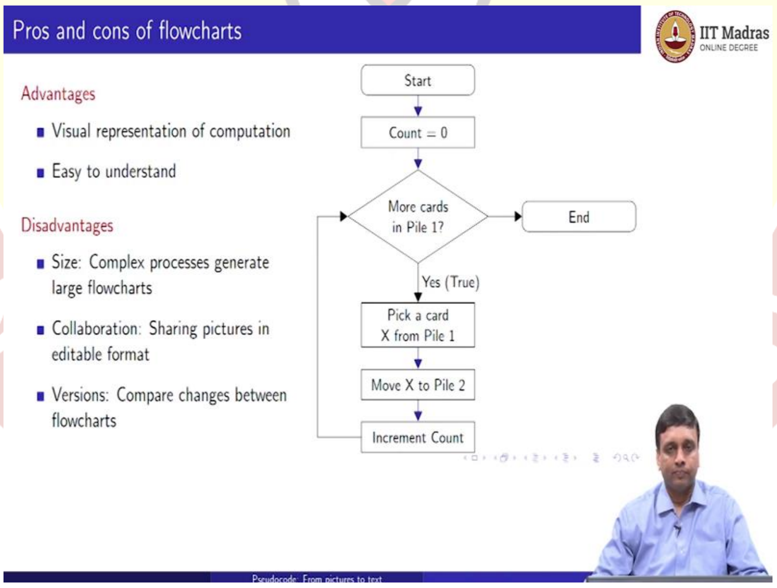
So, this flow chart has many different nodes or diagram elements in it. So, we have these nodes where something actually happens where computation happens where the count is initialized to 0 or we pick up a card or we move it to pile 2 or we incremented and these are called process nodes. And they are drawn as these rectangles with sharp corners.

On the other hand sometimes we have to take a decision about which way to go; do we have more cards or have we run out of cards. So, these diamond shaped boxes are called decision boxes and they allow us to change our direction in the flow.

We begin at the start node and we end at the end node. So, these are examples of what are called terminal nodes. So, these are nodes where the computation starts or ends. There could be many different ways of terminating a computation for example.

So, there may be more than one way of ending and finally, the arrows actually tell us how to go from one step to another. So, in most cases each step follows the next one. However, when we have these decision boxes for instance depending on the outcome of the decision, we may go one way or another. Here for yes, we go down and for no, we go to the right.

(Refer Slide Time: 02:08)



Pros and cons of flowcharts

Advantages

- Visual representation of computation
- Easy to understand

Disadvantages

- Size: Complex processes generate large flowcharts
- Collaboration: Sharing pictures in editable format
- Versions: Compare changes between flowcharts

So, now, why do we need to go beyond flowcharts? After all flowcharts have a nice pictorial view of the algorithm. So, we can visualize what is going on and they are very easy to understand once we realize what these diamond boxes and rectangular boxes mean.

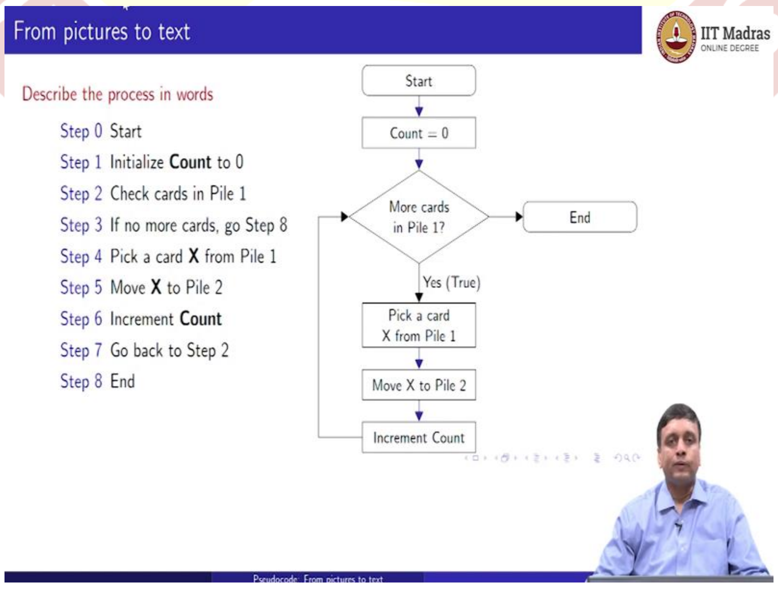
So, the problem is that most of the times when we are when we are constructing procedures, we are rarely working alone. We usually work in teams and as the size of the software grows as the procedure grows, so does the size of the flow chart and it becomes very hard to share this picture with somebody else in a format in which we can collaborate.

So, supposing I design part of the procedure and I send it to you and you are supposed to update it, then you need to be able to take my diagram and extend it without having to draw it again from scratch. So, just sending it as an image is not very useful, we need to have a format just like we do for documents.

So, supposing we are editing a document together, then you can take the document that already comes to you by email or by sharing on some kind of a shared folder. You can take the document and start working on the document without having to go back and retype everything. And we would like the same kind of flexibility when we are working collaboratively on designing procedures.

And of course, along with working collaboratively or even if you are developing it on your own, one thing that you have to keep in mind is that these documents or these flowcharts change. So, as we add things or modify things or make them better, we need to be able to compare what we have done. How does the current flowchart differ from the previous flowchart and this is not very easy to keep track of in this pictorial format.

(Refer Slide Time: 03:47)


From pictures to text

Describe the process in words

```
Step 0 Start
Step 1 Initialize Count to 0
Step 2 Check cards in Pile 1
Step 3 If no more cards, go Step 8
Step 4 Pick a card X from Pile 1
Step 5 Move X to Pile 2
Step 6 Increment Count
Step 7 Go back to Step 2
Step 8 End
```

So, instead of pictures, we will move to text. So, here is a very naive way of writing down the same procedure as text. So, we number the steps starting from the initial step 0 which is just to start the procedure.

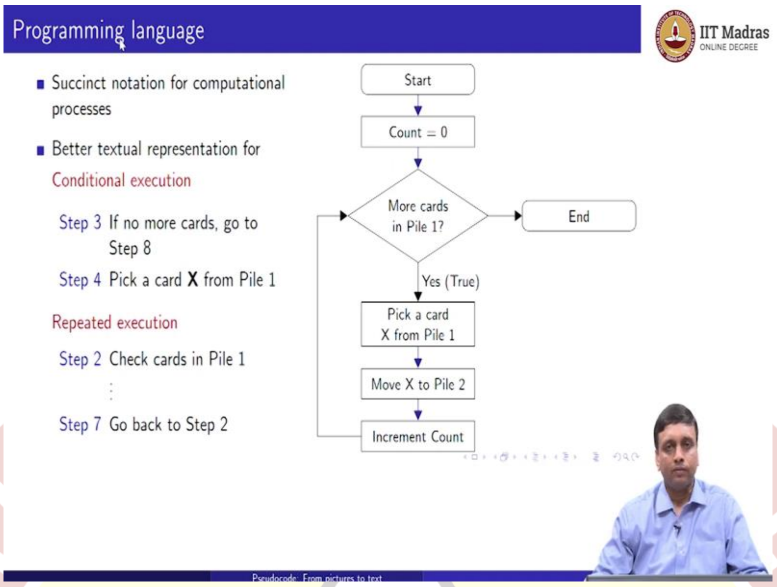
So, we just write down one step for every box more or less and the sequence of steps follows the arrows that are in the sequence. So, we have step 0 which is to start. Step 1 where we initialize the count to 0, step 2 is our decision box. So, we have to check the cards in Pile 1 and now we have to take a decision. So, now, we have to vary the flow and this is one problem when we have text because text is linear; you read it from top to bottom.

So, we have to have some way of describing the fact that there are two ways we can proceed and that is why we have these step numbers. So, it says if there are no more cards from step 3 instead of going to step 4 as you normally would; if there are no more cards, you directly skip ahead to step 8 which is End. Otherwise, if there are more cards, then you proceed. So, step 4 says pick up a card 6, step 5 says move it to Pile 2, step 6 says increment.

And now with respect to the flowchart, we have reached the bottom node and what happens after we increment count? Well, we have to go back and pick up the next card and for that we have to first check whether there are any more cards. So, we go back to the decision point which is step 2. So, we go back and check if there are more cards.

So, in this textual notation, we have used step numbers to label the steps and we use these step numbers in order to indicate when we want to divert from moving from one step to the next and instead move to a step which is either far in the future or far in the back.

(Refer Slide Time: 05:24)



Programming language

- Succinct notation for computational processes
- Better textual representation for

Conditional execution

Step 3 If no more cards, go to Step 8

Step 4 Pick a card **X** from Pile 1

Repeated execution

Step 2 Check cards in Pile 1

⋮

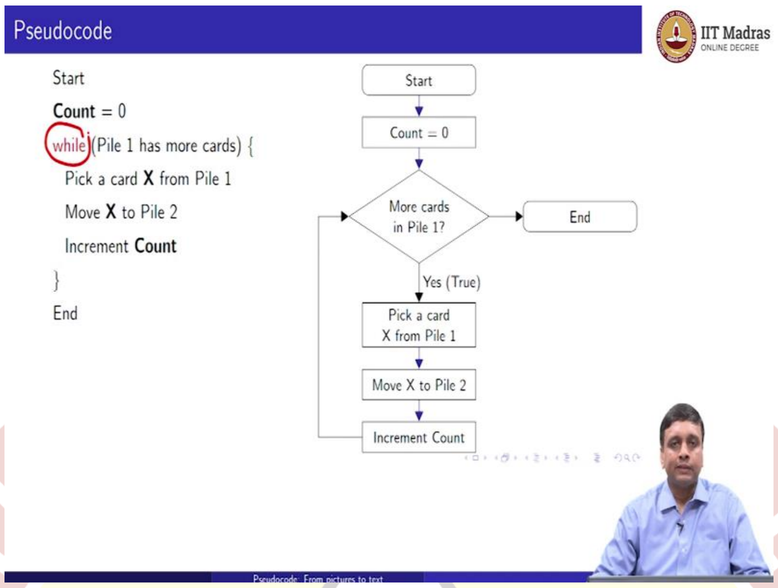
Step 7 Go back to Step 2

So, there is a better way of doing this and that is what we are going to call pseudo code and pseudo code is an informal way of describing what is called a programming language.

So, programming language is just a way of writing computational processes in text in a succinct way. So, you do not write lot of unnecessary things and some of the essential features of the process get captured in the text without being very cumbersome like we had the step numbers. So, for instance one of the things that we frequently need to do when we have computational processes is to do conditional execution.

For here in this example after step 3; if you go to step 4, it is only because step 3 has said no. So, step 4 is not always going to be executed and we do not we would like to represent it directly in the text somehow. That step 4 depends on step 3. Similarly, we have this repeated execution. The fact that we go up to step 7 and then come back to step 2. So, rather than having this explicit instruction to go back to a previous step, we would like to record that steps 2 to 6 represent something which has to be done and again and again.

(Refer Slide Time: 06:29)



Pseudocode

```
Start
Count = 0
while (Pile 1 has more cards) {
    Pick a card X from Pile 1
    Move X to Pile 2
    Increment Count
}
End
```

So, let us just jump into pseudo code. So, this is a pseudo code representation of the same procedure. So, we don't have any step numbers and we just follow the same principle that when we have a statement and a new statement of the next line, we will go from this statement to the next statement; however, we do have some variations.

So, now we have introduced here a special statement called while right. So, while is a statement which means what it says in English. So, when I say, while the lift is in motion, do not try to open the door. It means, so long as the lift is moving, do not try to open the door or while it is raining keep your umbrella open. So, while means, while some condition happens do something.

So, the first thing is to assign a value to a variable right. So, we begin with this Count variable and we assign it to 0. So, this is a basic statement in our notation for pseudo code.

The next thing is this conditional execution. So, we use the special word while and then we write the condition inside brackets just to make it clear. So, in this case the condition says Pile 1 has more cards.

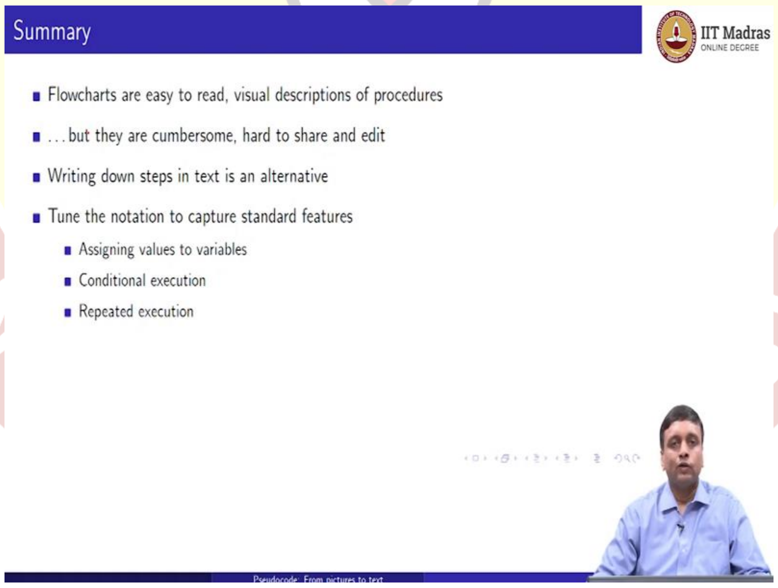
Now of course, this is a kind of informal statement, but it is precise enough that we understand what it means. This is what makes us different from real code. In a real programming language, the computer would have to know how to check this, but this is something which is kind of in between. So, we can get away with a few minor things like this.

So, we say while Pile 1 has more cards, we have to do something. So, while this condition holds, we have to do a number of things again and again until the condition becomes false.

How do we know which ones we have to do again? Well, we use these braces to indicate this block of steps which goes with the while. So, this says that the next three steps: picking up a card, moving it to the other pile and incrementing count. These are a unit which goes with a while and the fact that they are inside this brace ties it to the while.

So, we do not have to keep track unlike in the previous case where we had step 2, step 3 up to which step belongs to that condition right. So, these are some of the features that make a programming language or pseudo code more convenient to write procedures in a way that are easy to understand.

(Refer Slide Time: 08:38)



Summary

- Flowcharts are easy to read, visual descriptions of procedures
- . . . but they are cumbersome, hard to share and edit
- Writing down steps in text is an alternative
- Tune the notation to capture standard features
  - Assigning values to variables
  - Conditional execution
  - Repeated execution

IIT Madras
ONLINE DEGREE
Pseudocode: From pictures to text

So, to summarize we have seen that flowcharts are very easy to read and because they are pictorial and visual, they are very easy to understand. However, they grow very rapidly in size. So, they become very cumbersome to draw and represent on an even on a single sheet of paper or a single screen. They are very hard to collaborate with.

So, it is difficult to share a flow chart with somebody else in a format where they can continue to work on it. And as we said, it is also difficult to keep track of how one flowchart is different from the previous version. So, if you are keeping track of a process by which the computational procedure is getting refined, then we need to keep track of where the changes happened and how these changes happened and this is very difficult to do with a pictorial notation.

So, instead we decided to write down the steps as text and instead of just writing it down as step 1, step 2 and go to step 5 and so on; we tuned the notation to capture some standard features which we would normally expect in a computational procedure. The first one is assigning a value to a variable. This is a very basic thing because we keep track of variables all the time and we have to keep updating these values. So, assigning a value to a variable is a very basic step.

Conditional execution is also a very basic step. If some condition holds you do something otherwise, you do not do that thing. And repeated execution, you do something a number of times. You go through every card in the deck right. So, these are some of the features that we have seen.

And now we will look at more examples in the next lecture to understand exactly how to fit these to the flow charts that we have seen for the examples that we have done by hand in class.