Software Testing
Professor Meenakshi D'Souza
Department of Computer Science and Engineering
International Institute of Information Technology, Bangalore
Testing Source Code: Classical Coverage Criteria

(Refer Slide Time: 0:15)

Welcome back. This is the last module of this week, we pretty much come to the end of graph coverage testing for now at least we will revisit it later in the course, while we looked at control flow criteria for structural coverage, data flow criteria which includes data flow coverage, and then coverage based on call graphs, how to derive graphs from specifications, and what are finite state machines briefly, and how to get graphs. Graphs have been traditionally used in testing also and there are several traditional metrics that are available where graphs have been implicitly used specifically control flow graphs to test source code as a part of a software testing course it will help if you can get an idea of these traditional or classical coverage criteria to test source code.

(Refer Slide Time: 1:10)

Outline of this lecture

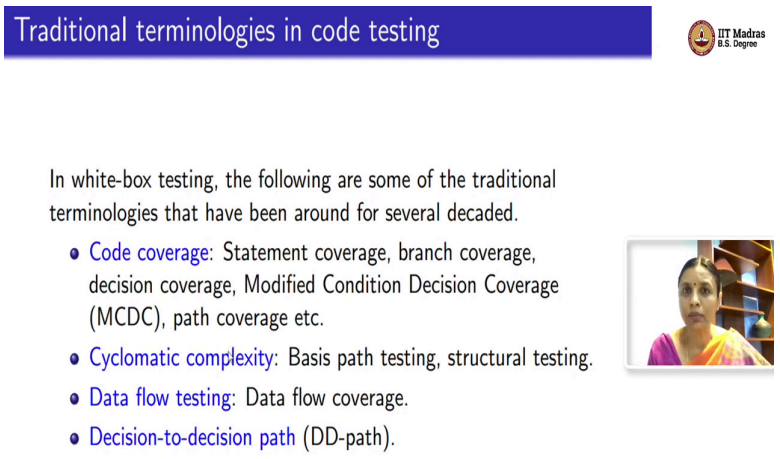
IIT Madras B.S. Degree

- Understand the traditional terminologies used for code testing, focussed on coverage.
- Understand how these are related to the graph coverage criteria that we have discussed till now.

Source code vs. graphs

IIT Madras B.S. Degree

- The most common graph model for source code is control flow graph.
- Structural coverage criteria over control flow graphs deal with covering the code in some way or other.
- Data flow graph: Augments control flow with data.
  - Data could be def and use of variables.
  - Data could be the actual values of variables.
- Other graphs: Inter-procedural call graph, execution order etc.



So, as I told you we will understand some old terminologies that have been there for several decades and how are they used for testing how are they specifically related to graph coverage criteria that we have seen till now. So, as I told you when you have source code, code within a function or method the most common graph model for source code is the control flow graph many IDs might generate control flow graphs automatically otherwise there are several open source tools that are available.

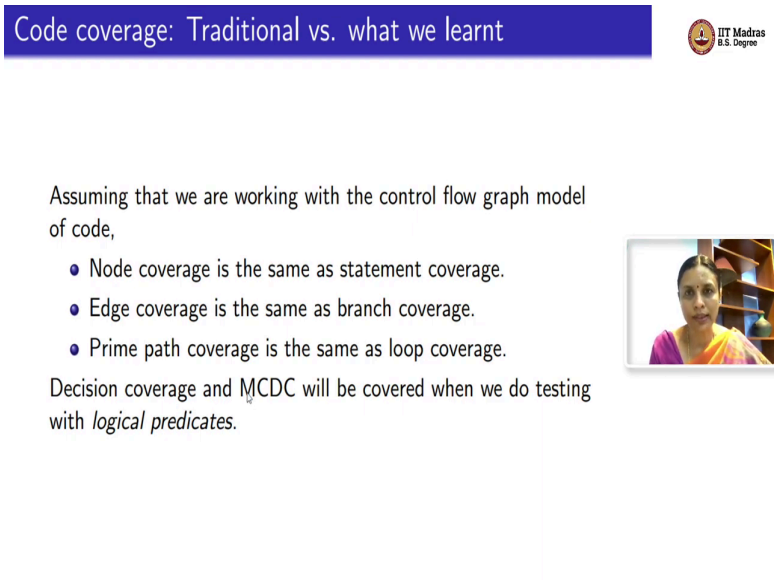
So, we saw structural coverage criteria over control flow graph covering the code in some way or the other then we augmented it with data definitions and uses and saw data flow coverage criteria then we saw call graphs sequencing constraints and so on. In traditional software engineering let us say 40 years ago or 50 years ago some standard terminologies were used. So, specifically for white-box testing of code these were the terminologies that were used the idea was to cover code in some way or the other code has statements branches the decisions that lead to branches when a code executes it goes through various paths.

So, I could have coverage criteria to execute statements, statement coverage execute every branch or every decision branch coverage and decision coverage might mean the same but they actually are slightly different we will see the difference as we go down the course. This is a specific version of decision coverage called MCDC which we will also see later and then path coverage, these have been traditionally used for testing along with that there is an important notion called Cyclomatic complexity which tells you one measure of how complex easy or difficult your code is in terms of the various execution possibilities that it can happen, and based on Cyclomatic complexity you can on the control flow graph generate and test for basis paths, and also test for some other kind of structure based on Cyclomatic complexity of

course data flow testing it remains the same data flow coverage which we have already seen and there is one more kind of paths that you can generate from control flow graphs they are called decision to decision paths abbreviated as DD-paths.

So, in this lecture we will see how the classical source code coverage criteria are related to the graph coverage criteria that we have seen what is Cyclomatic complexity how to calculate that what is Basis path testing how to get test requirements for basis paths based on Cyclomatic complexity and what is decision to decision paths or DD-paths data flow testing we have already covered.

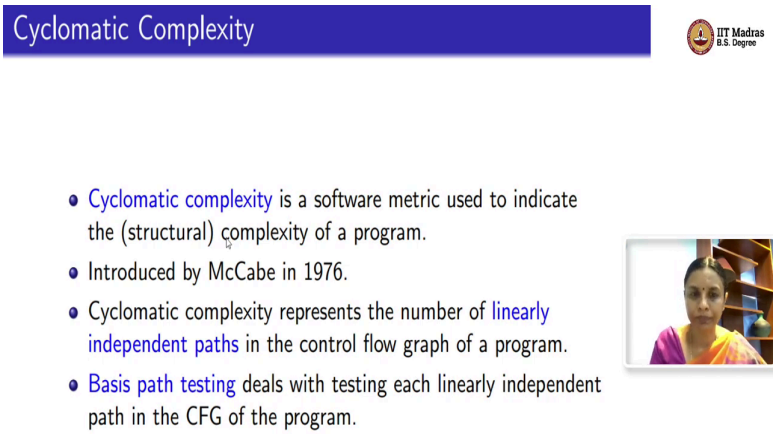
(Refer Slide Time: 4:06)



So, assuming that we are working with control flow graph model of a method or of a code you can always trace between individual statements in the code and nodes in the control flow graph. So, in the structural coverage if you remember we saw Node coverage, Edge coverage, Prime path coverage along with a few other things Node coverage is the same as statement coverage if I, if I if my test requirement says execute every node it means execute every statement if my test requirement says execute every Edge it means execute every Branch.

Similarly, Prime path coverage is the same as Loop coverage if you go back here there was also decision coverage MCDC. So, these are not directly related to control flow graphs decision coverage is related to branch coverage but a slightly more elaborate version of that. So, we will see these two separately in the next week's module onwards we are going to begin a new module from week five where we are going to learn logic based testing.

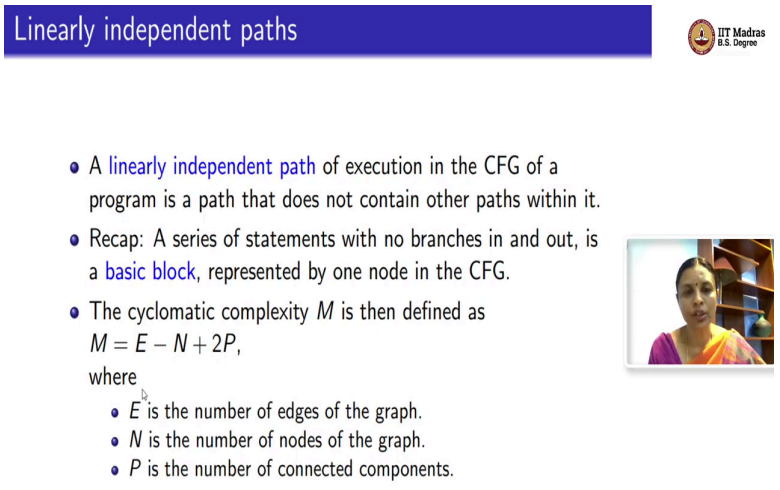
(Refer Slide Time: 5:19)

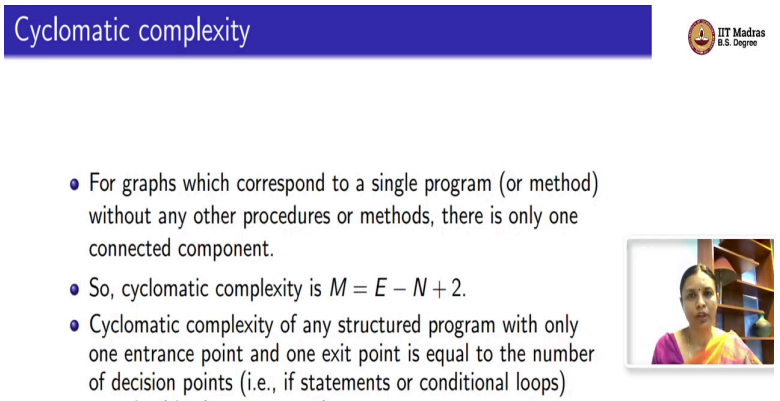


So, we're going to see decision coverage and what is MCDC next week. Another popular term that has existed in software testing in fact it was introduced in the year 1976 by McCabe is the notion of Cyclomatic complexity. So, what is it say it is basically one standard quality metric that is used to indicate how complex a software is in terms of its structure specifically branching structure like in the control flow graph how what is easy what is difficult. Suppose there are lots of branches out of a particular node then the software is considered more complex as against lesser branches out of a particular node.

So, Cyclomatic complexity is a measure of how complex a piece of software is it represents the number of linearly independent paths in the control flow graph of a program. So, we will very soon see what a linearly independent path is but Cyclomatic complexity says take the control flow graph and count the number of linearly independent paths in the graph that is what is Cyclomatic complexity when my TR becomes test for each linearly independent path in the CFG of a program then we get what we call basis path testing.

(Refer Slide Time: 6:38)





So, what are linearly independent paths take the control flow graph of a program one point to note here is while we are looking at control flow graph of a program we you remember we had this notion of a basic block where if I have a continuous sequence of statements without any branching in them I collapse those continuous sequence of statements into one node. So, that notion of a basic block is very important here you may not have freedom to keep two three nodes corresponding to different, different groups of basic blocks in the CFG all the contiguously occurring statements must occur as one basic block in the CFG one node in the basic block of the CFG.

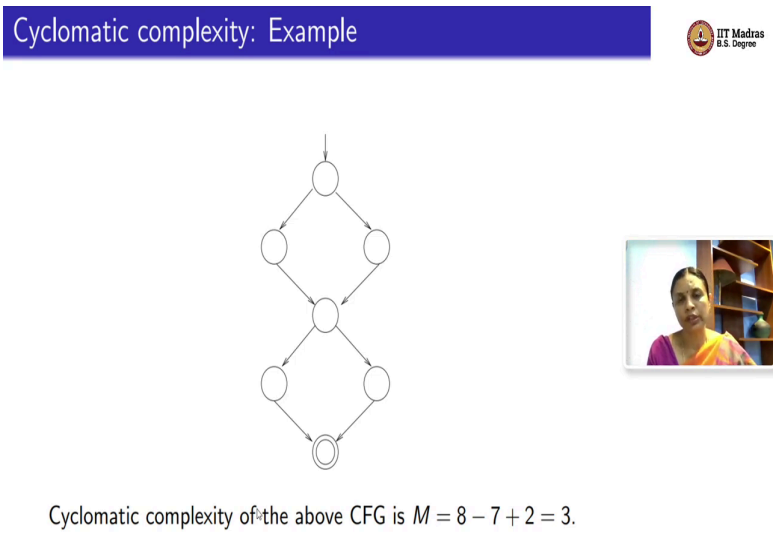
So, once I have a CFG like that a linearly independent path in that control flow graph is a path that does not contain other paths within it. You might want to pause here for a second

and compare how similar it is to that of a prime path you remember Prime Paths, Prime Paths where simple paths that do not contain other paths within it. So, linearly independent paths are very similar to Prime Paths in fact every linearly independent path is also a prime path.

So, once I have a CFG and I compute these linearly independent paths we know how to compute them we saw algorithms for computing that Cyclomatic complexity is defined by this number it is just a number. So, it says count the number of edges in the graph. Let it be E count the number of nodes or vertices in the graph let it be N and count the number of connected components in the graph let it be P. If you remember in the first before we started graph based testing I had introduced several basic terminologies and graphs and connected components was one such thing that we saw.

So, Cyclomatic complexity has a number is given by this formula E minus N plus 2P where E is the number of edges N is the number of nodes P is the number of connected components. Sometimes the as we saw control flow graph can just correspond to a single program the single program can be a method as on its own without any other procedures any other method. So, there is only one connected component there are no bits and pieces that are hanging there. So, Cyclomatic complexity is you this P becomes one in this formula. So, Cyclomatic complexity is nothing but E minus N Plus 2. So, Cyclomatic complexity of any structured program with only one entry point and one exit point is roughly equivalent to the number of decision points that you get in the control flow graph.

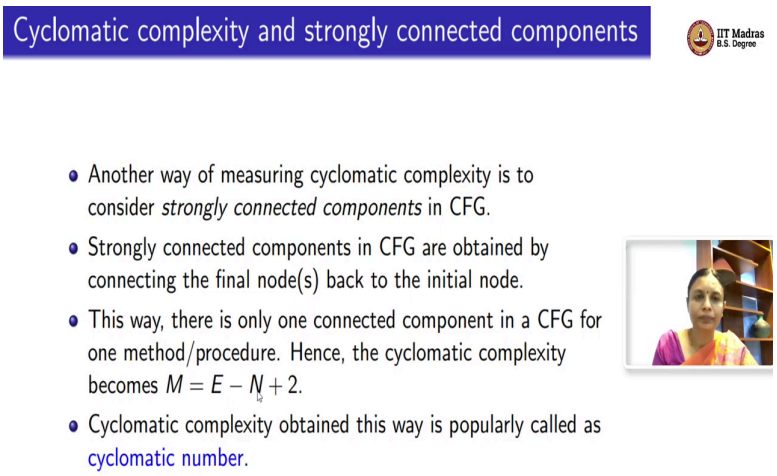
(Refer Slide Time: 9:20)



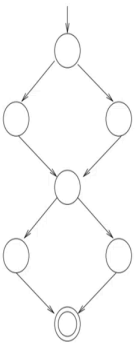
So, here is an example let us say the control flow graph looks like this. So, if you see here this top node here is the initial node, this last node here is a final node, initial node happens to have a decision point two branches coming out this node also happens to be a decision point two branches coming out. So, Cyclomatic complexity is this graph has 8 edges 4 for the top diamond 4 for the bottom diamond 7 nodes and it has only one connected component. So, E minus N plus 2P becomes 3.

So, this graph has Cyclomatic complexity three roughly corresponds to this the paths that go out of this decision points plus the paths that go out of this decision points. In general, this graph has two decision points the Cyclomatic complexity based on which notation you follow could be plus or minus 1 with reference to the number of decision points the number of decision points plus 1 or the number of decision points minus 1.

(Refer Slide Time: 10:22)



Cyclomatic complexity: Example



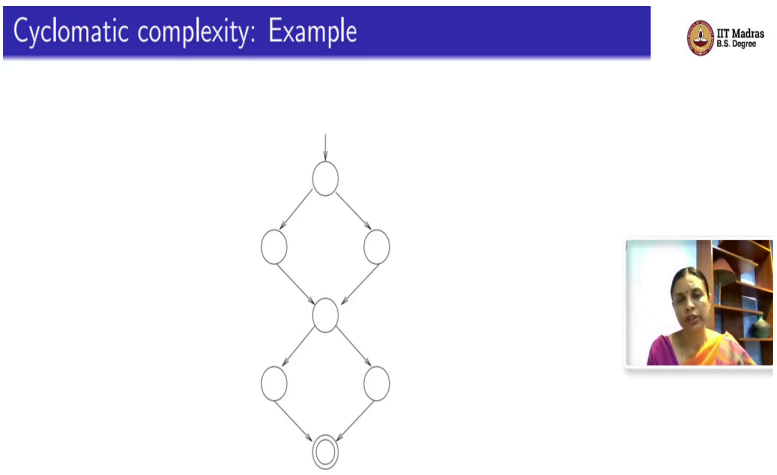
Cyclomatic complexity of the above CFG is $M = 8 - 7 + 2 = 3$.

So, one more point about Cyclomatic complexity versus strongly connected components another way of measuring Cyclomatic complexity is to consider strongly connected components in the control flow graph. We again saw how to get strongly connected components by using depth first search strongly connected components are obtained by taking a control flow graph like this simple way to get it as one connected component is to put a back edge here. Then the whole thing becomes strongly connected in which case cyclo you can get rid of the P factor and the Cyclomatic complexity becomes just E minus N Plus 2. So, sometimes because Cyclomatic complexity is just a number, Cyclomatic complexity is also called Cyclomatic number and it is believed that if it is a number less than 10 then the graph the code is not too complex in nature the code can be handled that is what it means.

(Refer Slide Time: 11:19)

Basis path testing

- Basis path testing is used to test a code with respect to its cyclomatic complexity.
- The CFG is analyzed to find a set of linearly independent paths of execution. This is the TR for basis path testing.
- Test paths for such a TR can be the linearly independent paths themselves or any test path that contain these paths.
- Basis path testing subsumes branch coverage. Complete path coverage subsumes basis path testing.



Based on Cyclomatic complexity I can do what is called basis path testing. So, as I told you the control flow graph is first analyzed to find a set of linearly independent paths for execution. We saw an algorithm for this, the algorithm that will generate Prime Paths can be slightly tweaked to get the set of all linearly independent paths that is your test requirement for basis path testing, test paths for such a test requirement that is for basis path testing can all be linearly independent paths themselves or test paths that begin at the initial node end at a final node and contain one linearly independent path within it that is what can be a test path and as we saw basis path testing obviously includes the decision points because Cyclomatic complexity is all about decision points.

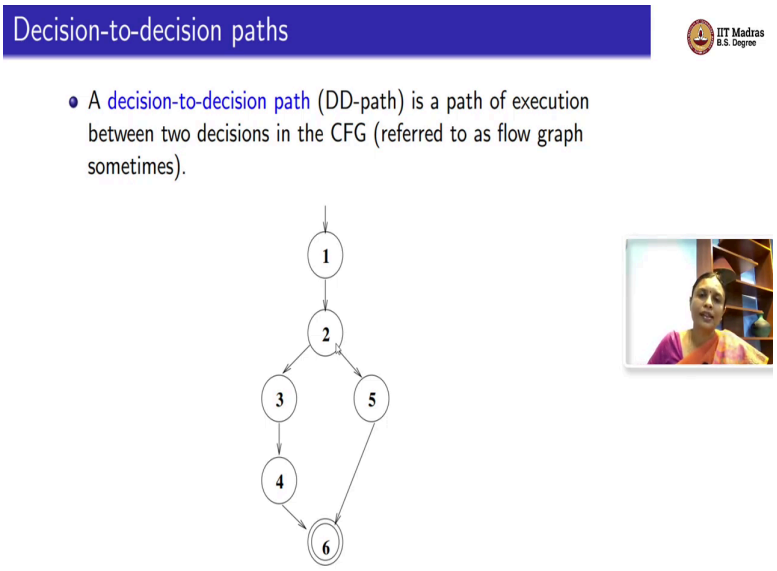
So, it subsumes edge coverage or branch coverage obviously this represents only linearly independent paths. So, what subsumes this complete path coverage trivially subsumes this because complete path coverage is about testing all paths even though it could be infeasible quite often. So, that was basis path testing I hope you understood what it is.

So, in summary take a control flow graph like this this control flow graph is likely to have decision points in these decision points you try to measure. So, here for example the number of linearly independent paths could be you go start from here you go left then you come to the center you stay left and then you come here or you start from the top go right come to the center stay right and come back and then or you could do zigzag start from the top go left come to the center go right end start go right come to the center go left and end.

So, there is a one-to-one correspondence between Cyclomatic complexity and linearly independent paths this way the number could be plus or minus 1. these terms have not been

very robustly defined that is why the kind of graph based testing that we saw is more robust than kind of the kinds of concepts like this which have been older but not very well defined but it helps to know what a Cyclomatic complexity is and the fact that if it is less than 10 then it is considered useful for testing. And the paths the kind of paths that you consider for testing Cyclomatic complexity are linearly independent paths and if I test with my test requirement as the set of linearly independent paths it is called basis path testing.

(Refer Slide Time: 13:57)

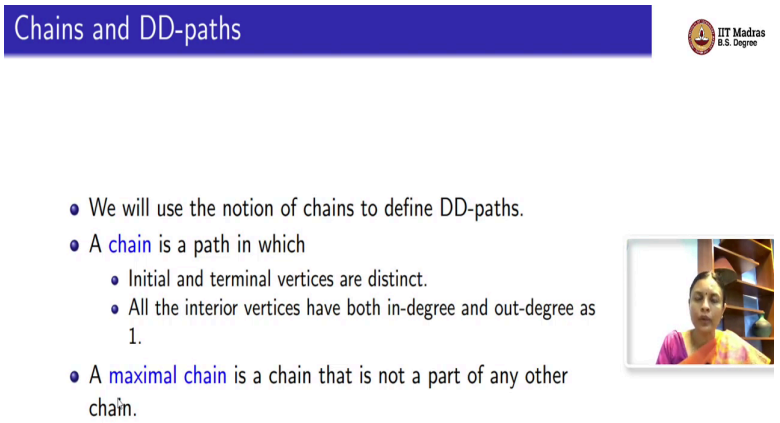


The next comes decision to decision paths as I told you earlier or DD-path. So, here how do we look at it. So, if we again take the control flow graph same way sometimes when you look at the literature for DD-paths they might refer to control flow graph as flow graphs if you refer to certain textbooks or the internet but it is one and the same. So, you take a control flow graph look at all the decision points in the control flow graph.

So, if I take this example here which is the decision point in this control flow graph node 2, there is a decision there is a branch from 2 to 3, there is a branch from 2 to 5. a decision to decision path basically eliminates this decision and consider paths from the initial node that lead to the decision points and the paths that come out of the decision points into maybe let us say the next decision point or go to the end. In this example we had only one decision point that is a 2 and the two decisions that come out of two are 2 to 3 and 2 to 5.

So, it is like take this 2 the edge 2 3 edge 2 5 keep it aside right. So, then then take the paths from the initial node that lead to the decision what would be that 1 to 2, then take the paths that go out of the decision that would be 3 to 4 5 to 6 of 4 to 6. So, a decision to decision path is take the control flow graph consider all the decisions in the control flow graph in this example we have only one keep them aside and then take the paths that lead you from one decision to the other one decision to the other in the CFG bit by bit and that is what is called a decision to decision path.

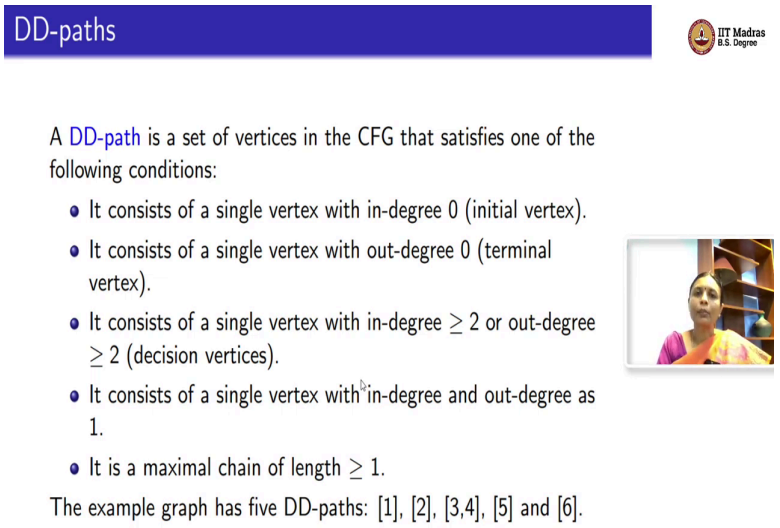
(Refer Slide Time: 15:54)

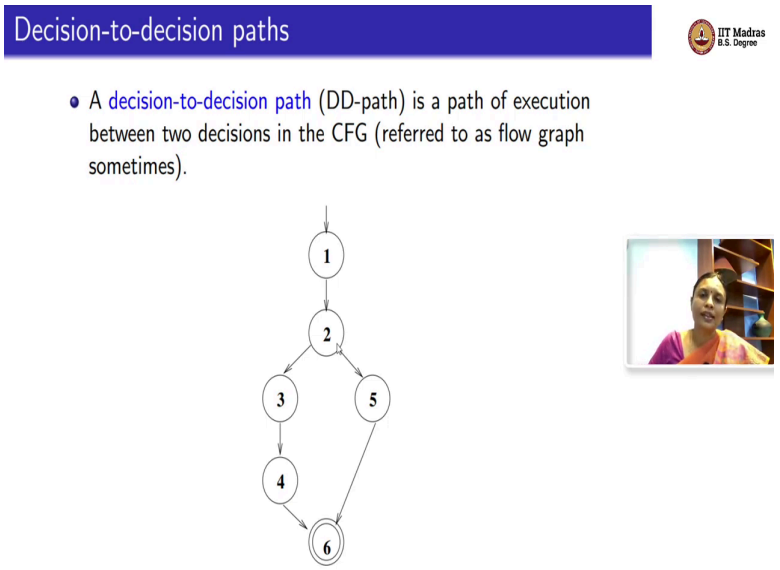


So, for formally defining a decision to decision path we need the notion of a chain. What is a chain? A chain is any path in which the initial and terminal vertices are distinct they do not come back like in a loop all the interior vertices have both in degree and out degree as 1 is that clear chain is a path where initial vertex and terminal vertex are distinct and along the way all the interior vertices.

The vertices that come along the path which are not initial and not final they are in degree and their out degree both will be 1. If you forget the notions of what in degree and out degree are go back and recap the first set of slides on graphs a chain can be maximal a maximal chain is a chain that is not a part of any other chain like we had in prime paths this is it can it cannot contain any other chain within itself it is the maximal possible chain.

(Refer Slide Time: 16:59)



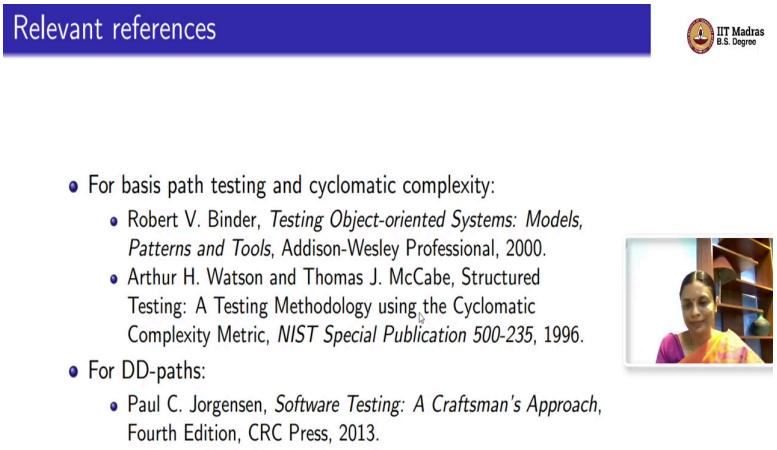


So, now we are ready to define DD-paths. A DD-paths is a set of vertices in the control flow graph that satisfies one of the following condition. One is to be highlighted and remember it is in the in the CFG that satisfies one of the following conditions. It consists of a single vertex of in degree 0 which is for our case the initial vertex in the graph, it consists of a single vertex of out degree 0 which in our case is the terminal vertex, the final vertex in the graph or it can consist of a single vertex within degree greater than or equal to 2 or out degree greater than or equal to 2 which are basically your decision vertices like we saw to the vertex 2 in that example or it consists of a single vertex within degree and out degree as 1 or it is a maximal chain of length greater than or equal to 1.

It might sound like this is a complex definition it has five paths but remember it is easy to implement it and one way to interpret it as I told you is take a control flow graph take all the decisions keep this decision edges away then move from one decision to the other decision keeping this notion of maximal change in mind that is what is written out here in terms of a enumerative definition of a DD-path for this example the DD-paths are 1, 2 to 3, 2 to 5 is the decision.

So, we keep it aside then 3 4 is a DD-path because it comes out of a decision 4 to 6 and 5 to 6 lead into 6. So, just 6 is a DD-path because it is at the final node and 5 is a DD-path. So, that is what is given here 1 2 paths consisting of the single vertices 1, 2, 5 and 6 and path 3, 4 that goes out of the left branch of this decision. So, 1, 2, 5 and 6 and path 3 4. It just leaves out these two edges that correspond to the decision 2 to 3, 2 to 5 is that clear that is what is a DD-path these may not be very important we may not use it too much in our course but the purpose of this lecture was to tell you that traditionally apart from the graph coverage criteria that we saw there is also Cyclomatic complexity there are basis paths and there are DD-paths.

(Refer Slide Time: 19:40)



So, keep them in mind and if somebody asks you what it is this would be a useful way of referring to them. So, the material that I taught you about where is DD-Paths you could refer to it in this book by Jorgensen and for basis path testing and Cyclomatic complexity there are some classical books that are available and this is the makeup is original paper that defines I mean article that defines Cyclomatic complexity. So, I will stop here I will see you next week where we start a new module. Thank you.