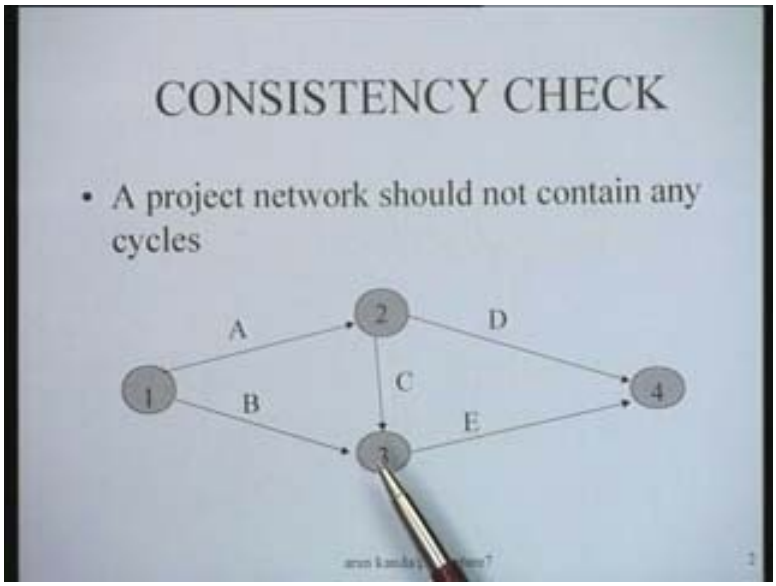
Project and Production Management
Prof. Arun Kanda
Department of Mechanical Engineering
Indian Institute of Technology, Delhi

Lecture - 8
Consistency and Redundancy in Project networks

In today's lecture we are going to be talking about development of the project network especially with regard to consistency and redundancy in the project network. As I had indicated in my last lecture it's necessary that the network should be consistent and also it is desirable that there should be no redundancies in the project network. In today's lecture we are going to talk about procedures that are used for checking consistency and removing redundancy in the project network. Let us recall what we mean by consistency. Consistency check broadly means that any project network should not contain any cycles. This small network is an example of a consistent network because it does not contain any loops.

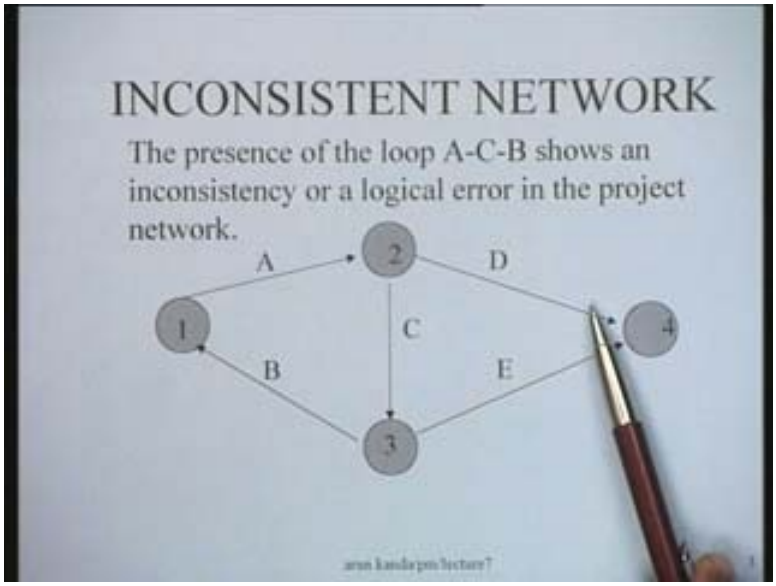
(Refer Slide Time: 2:04)



All the arrows are so directed that they go and define various paths without defining any loop. You would recall that the presence of a loop in a project network indicates an inconsistency because it implies that the activity should be completed before it has started and that is the logical inconsistency which comes up and therefore we must not have any inconsistencies in the project network. If you look at this network it looks very similar to the previous network. But this particular network is a network which contains a logical inconsistency because of this loop A C B. There is something wrong in the direction of the arrow here. But in a large network detecting these kinds of inconsistencies could be quite a formidable task but they must be detected. These errors must be detected because

they are logical errors and they would produce a wrong analysis if we rely on that. In fact a proper network analysis would not be possible on an inconsistent network.

(Refer Slide Time: 3:31)

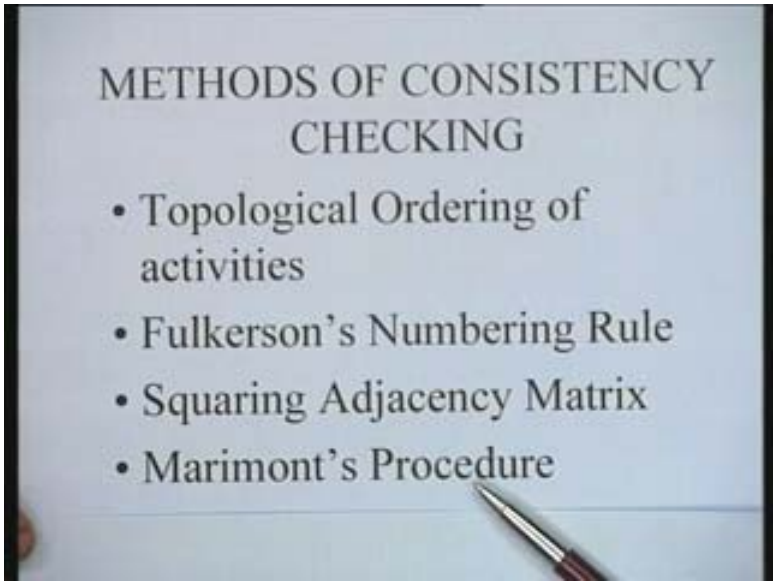


What are the various methods for checking inconsistency in project networks? We will briefly examine some of the commonly used methods for checking consistency. The first method of checking consistency in project networks is the process of ordering the activities in topological order as they say. Topological ordering of activities is a method which is perhaps the most popular method for checking consistency and in essence this method tries to make sure that the activities are ordered in a list such that no activity appears until all its predecessors have appeared. That is the essence of the procedure.

The second method for checking consistency of project networks is Fulkerson's numbering rule. Fulkerson's numbering rule is actually a procedure by which we take a network and number the nodes of the network in such a manner that for every activity going from node i to node j, the node number i is strictly less than node number j. If you can accomplish this numbering it shows that there is no inconsistency in the network. This is again a very effective and a very simple procedure for checking consistency of project networks. The third procedure which we are going to be talking about today is the procedure of squaring the adjacency matrix. The adjacency matrix for a project is a matrix representation of the project. Rather than representing the project as a network it can also be represented as various matrices. The adjacency matrix is one representation of the project network and by performing certain operation on the adjacency matrix we can check whether the matrix or the project network is consistent or not.

We will examine these procedures in detail and the fourth procedure that we might talk about is Marimont's procedure which can be applied equally effectively on either the adjacency matrix or the network itself and this method can also reveal any inconsistencies which are present in the project network.

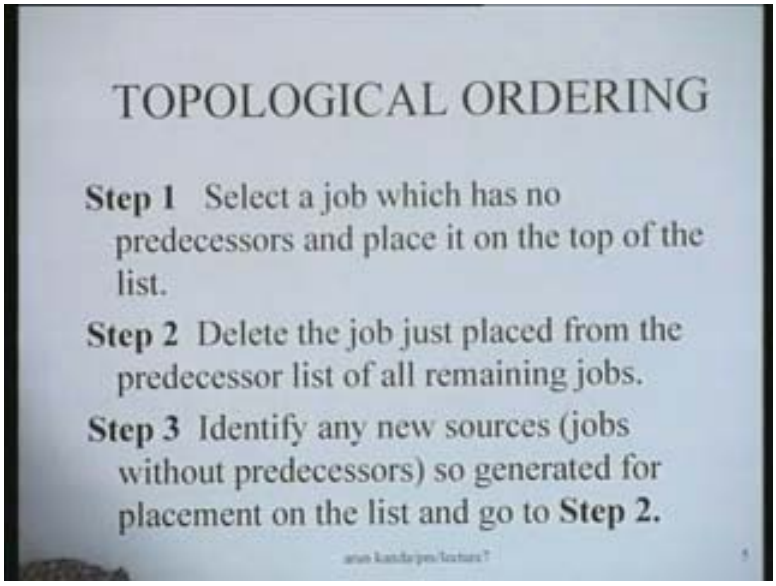
(Refer Slide Time: 6:19)



Different methods have their different range of applications. Topological ordering is done when you are dealing with activity lists and you are trying to topologically order the jobs. Fulkerson's numbering rule is most suitable when you have drawn the network in an A-O-A mode or an A-O-N mode and you are trying to number the jobs or number the nodes in such a manner that this is satisfied. So that stage it's useful. If you are not dealing with either the list or the project network but you are dealing with the adjacency matrix of the graph then this third method is very useful and the fourth method can be used on both the adjacency matrix as well as the network as we have seen.

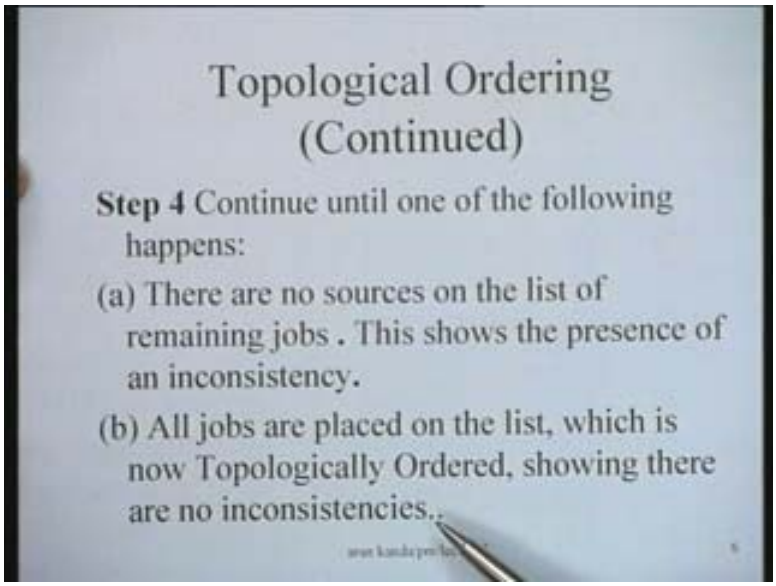
The procedure of topological ordering of activities can be summarized as follows. The first step in the procedure is to select a job which has no predecessors and place it on the top of the list. Such a job is a source which has no predecessors. It's a source in the network or otherwise it's a job which has no predecessors. You identify such a job and place it first on top of the list. Step 2 is delete the job just placed from the predecessor list of all remaining jobs. The job which has just been placed is deleted from the predecessor list of all the remaining jobs. In this process you can hopefully generate some new sources because that job has been deleted from the predecessor list of the jobs. In step 3 what you try to do is you identify any new sources, that is jobs without predecessors so generated for placement on the list and then go back to step 2. You keep on applying these steps in an iterative fashion till one of the two things happens.

(Refer Slide Time: 8:32)



You continue till there are no sources on the list of the remaining jobs. There are no sources but the list is not yet empty. There is a list of remaining jobs but there are no sources. If this happens this shows the presence of an inconsistency. On the other hand if all the jobs are placed on the list which are now topologically ordered showing there are no inconsistencies.

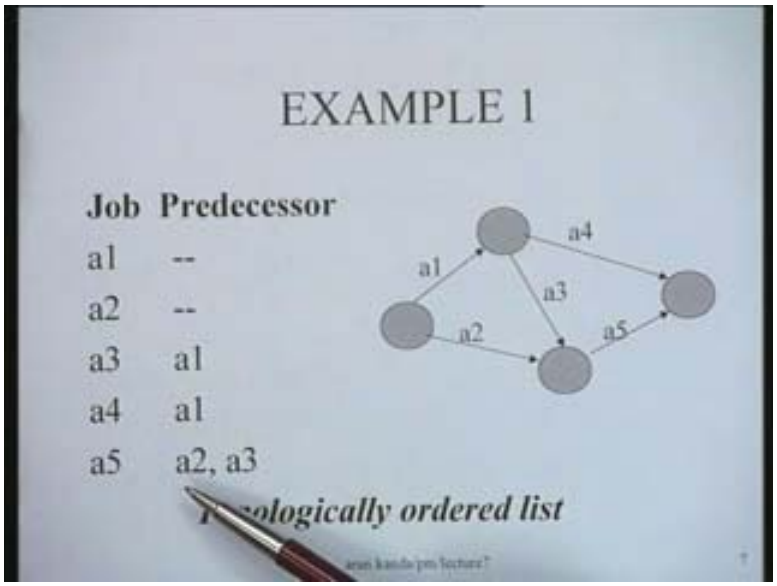
(Refer Slide Time: 9:05)



If there are no jobs remaining you are able to place all the jobs on the list, it clearly shows that you have been able to topologically order these various jobs. Let's take a couple of examples to illustrate this procedure. Take for instance a very small example of this

nature in which we are given the jobs and the predecessor list and in this list a1 is the job without any predecessors. a2 is a job without any predecessors. Then a3 comes on the scene and its predecessor has already appeared on the list. a4 comes on the scene; its predecessor has already appeared on the list and then a5 comes with a2 and a3. We have been able to topologically order the jobs. This is a topologically ordered list.

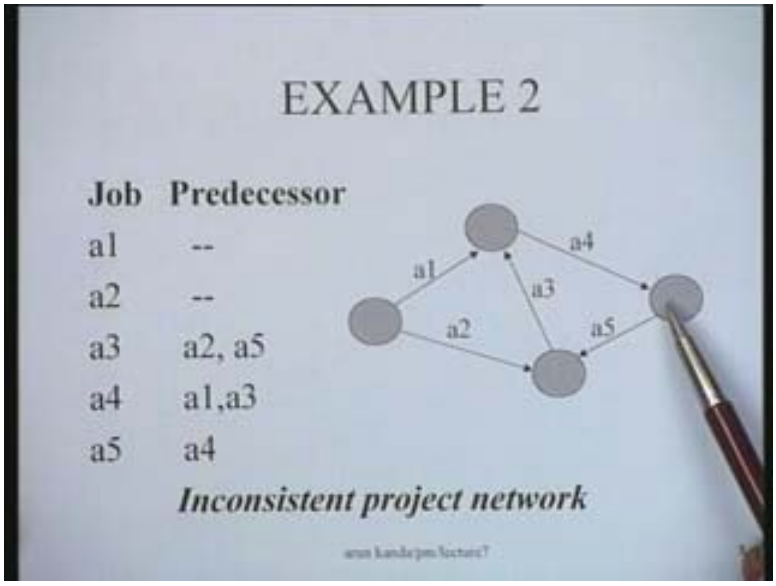
(Refer Slide Time: 9:57)



It shows that the network is consistent and you can see that the project network is actually a network which is something like this. It has no loops.

Let us now take an example of an inconsistent project network. Look at this example. Look at the list first. We are talking only about the list to begin with. Job a1 has no predecessors; a2 has no predecessors. So this is on the list. Then a_3 is placed. It has predecessor's a2 and a5 but a5 has not been placed on the list. It cannot be topologically ordered. Similarly a4 has a1 and a3. Both a1 and a3 have been placed. This is in the correct order. You can find that there is no way that you can topologically order this list. You cannot place this list in topological order and if you draw the graph of this you find that in this case there is a well defined loop here. This is an inconsistency.

(Refer Slide Time: 11:06)

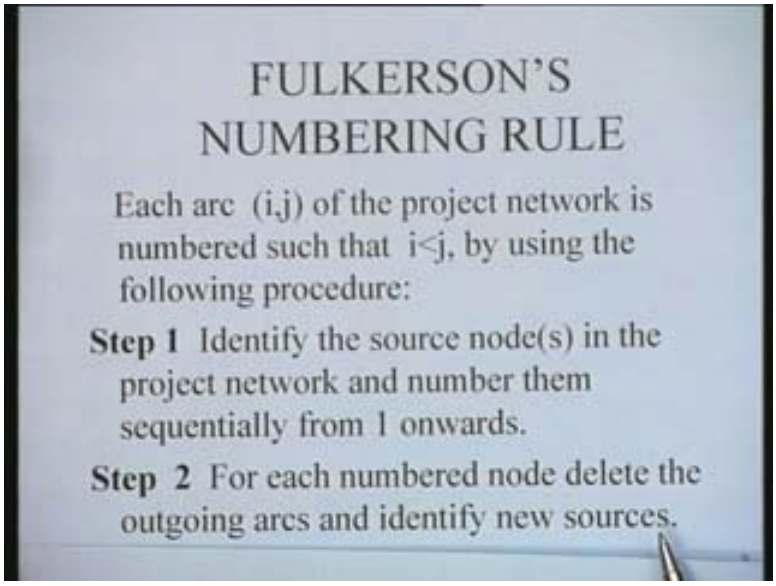


The inconsistency can be revealed by the simple fact that we are not able to topologically order the list by means of the procedure which we just identified and therefore this is an inconsistent network.

Let us now examine Fulkerson's numbering rule. Fulkerson's numbering rule is as I indicated to you essentially a procedure by which we are going to number the nodes of the network. It's a very convenient procedure and the essence of the procedure is that each arc (i j) of the project network is numbered such that i is strictly less than j. When you are referring to a particular job you will always be going from a lower numbered arc to a higher numbered arc and not vice versa and the procedure that is adopted for doing this numbering is as follows. The first step of the procedure is to identify the source node or nodes in the project network and number them sequentially from 1 onwards. If there are 3 nodes in the network which are sources you can call them 1, 2 and 3 arbitrarily in any order. It doesn't matter.

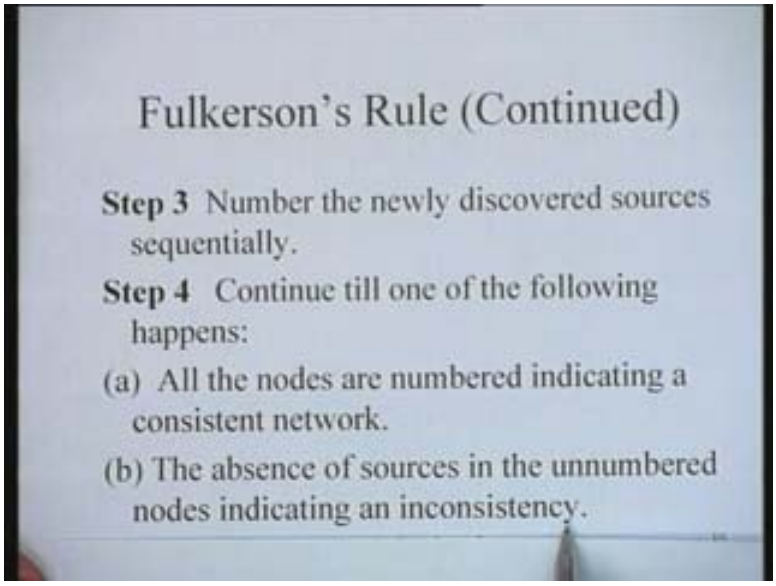
Having identified these sources what you do next is for each numbered node you delete the outgoing arcs and identify new sources.

(Refer Slide Time: 12:43)



From the arc, from that particular node which has just been numbered delete the arcs which are going out and in the process identify new sources which are generated by this particular procedure. What you do next is you number the newly discovered sources sequentially again. Whatever numbering you go ahead and number the newly discovered sources sequentially and continue until one of the following things happens: all the nodes are numbered indicating a consistent network. So consistency in the network is indicated by the ability to complete the numbering of all the nodes in this particular manner. The absence of sources in the unnumbered nodes; if there are some nodes which are not yet numbered and you were not able to find a source among them then it indicates an inconsistency.

(Refer Slide Time: 13:53)



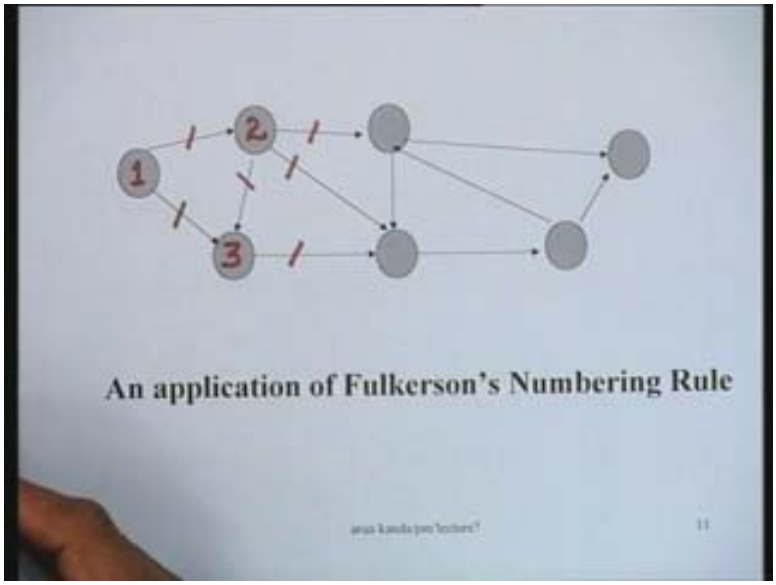
We cannot really number the nodes in that fashion. This is the inconsistency. To illustrate the procedure let's look at an example. Here we have a network and in this network the nodes are not yet numbered. We want to find out whether this is a consistent network or it is not a consistent network and we would like to see whether we can number the nodes according to Fulkerson's numbering rule. We look at the network and identify the source. This node is the only source in the network because it has both arcs which are going out. There is no other node in the network which has this property. We can call this as node number 1 and once this node is numbered as 1 we can delete the two arcs which emanate from this particular node and once we delete these two arcs which emanate from this node we can identify whether any new sources have been generated.

Have they been generated? This node is not a source yet. But this particular node is a source because this node arc has been deleted and these 3 arcs are there. These are the only 3 arcs which are coming out of this node. I can number this arc and I can give it the next number. I can call this as node number 2 and once I number this node as node number 2 I can by the same process, delete these 3 arcs which emanate from node number 2. The second node which is numbered I can delete these 3 arcs now which emanate from node number 2. Once I do that I try to identify if there are any new nodes which are generated, new sources which are generated. This node is not a source. This node is not a source because it has arcs coming in and going out but this one is a source. So I can number this as node 3 and once I number this node as number 3 I can delete the arc which emanates from this particular node and having deleted this arc now let us try to find out if there are any sources.

What do you find? This is not a source. This node is not a source. Why because this arc has been deleted. This arc but one incoming arc is there and one outgoing arc is there so this not a source. This is also not a source. Is this a source? It has 1 incoming arc, 2 outgoing arcs. It is not a source. We have reached a stage where we are not able to

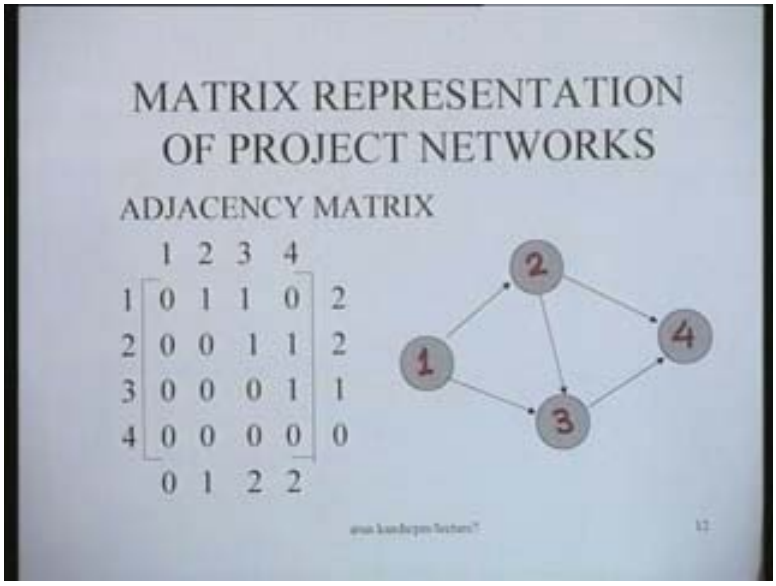
number any of the remaining nodes the simple reason that there are no sources among them. We have been able to number only 3 nodes in this network and the remaining nodes are not numbered and you can see actually that the numbering rules stops here because we cannot number and this shows that the network is actually inconsistent. A closer look at the network actually shows that there is a presence of a loop here; this, this and this is a loop. That portion of the network which has this loop did not allow the numbering procedure to proceed any further and therefore we have been able to detect the inconsistency in the network. Had the network been consistent we could have continued with the process and would have been able to number all the nodes.

(Refer Slide Time: 17:52)



This is an application of Fulkerson's numbering rule. Let us now look at the matrix representation of projects. In many computer applications and in other applications it's not the project network which can be stored as such in the computer but it is stored conveniently as a matrix. Quite often we have to work on matrices of the project. We look at some of the common matrix representations of projects. Probably one of the most convenient matrices to represent a project is known as an adjacency matrix. For instance for this particular network what we notice is that we can number this network according to Fulkerson's rule.

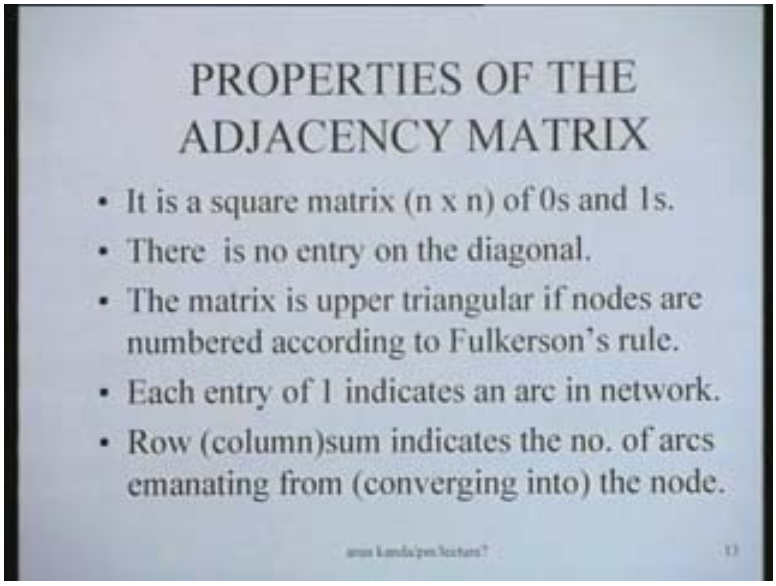
(Refer Slide Time: 19:04)



Each arch is now numbered such that i is less than j. This is not necessary when we have to draw up an adjacency matrix. The adjacency matrix of this particular network is defined as follows. It's a square matrix whose size is n by n where n is the number of nodes. In this graph there are 4 nodes; 4 by 4 matrix. It is very much like a, from to chart. From 1 there are 2 arcs to 2 and 3. Other elements are zero. From 2 there are 2 arcs to 3 and 4. These are shown here. From 3 there is only 1 arc to 4 and from 4 there are no arcs. It's a matrix of zeros and ones in which you find that the ones indicate arcs. In fact the number of ones is equal to the number of arcs in the network. There are in this case 5 ones. These correspond to the 5 arcs which exist in this network.

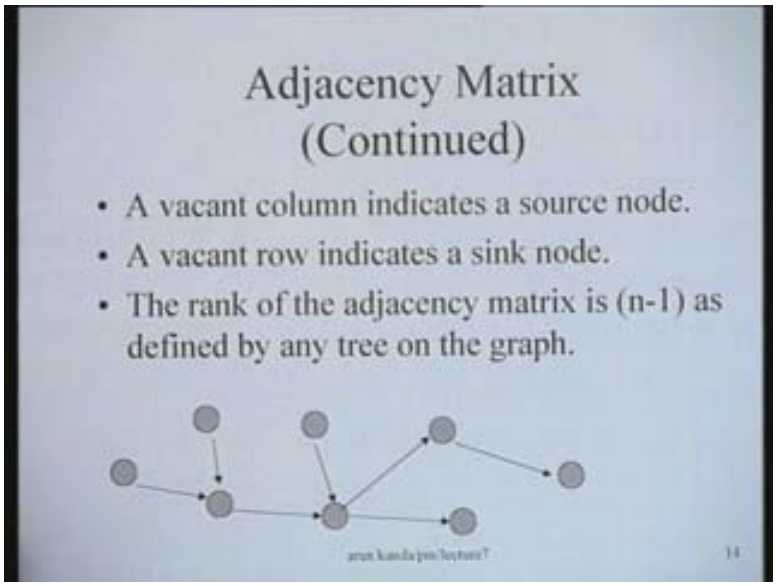
If you take the row sum or the column sum, the row sum for instance shows you that from node 1 there are 2 outgoing arcs. From node 1 there are two outgoing arcs. If you take the column sum for instance, it shows that onto node 3 there are 2 incoming arcs. Onto node 3 there are 2 incoming arcs. Rather than dealing with the network you can deal with the matrix only and it gives you the entire structural information about the project network. This is the adjacency matrix. The adjacency matrix has certain very interesting properties. Some of the properties of the adjacency matrix are summarized here. It's a square matrix of size n by n consisting of zeros and ones; that we have seen. There is no entry on the diagonal of the matrix. Why is that so? Because there is no self loop; that is there is no arc which goes from the node to itself. That is why there is no entry on the diagonal. The matrix is upper triangular if nodes are numbered according to Fulkerson's rule. This is what happened in the example that we considered. Each entry of 1 indicates an arc in the network. The row sum indicates the number of arcs emanating from the node and similarly the column sum indicates the number of arcs converging into the node as we just saw.

(Refer Slide Time: 21:49)



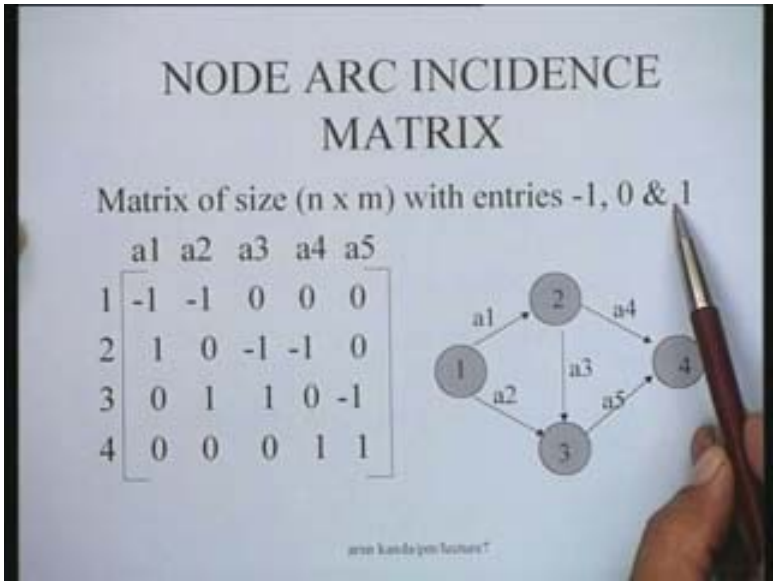
Another interesting thing about the adjacency matrix is that it's very easy to identify the source and sinks. How? A vacant column will indicate a source node. A vacant row will indicate a sink node. The rank of the adjacency matrix is n-1 as defined by any tree on the graph. Any tree on a graph of n nodes will have n-1 arcs just as this particular tree is there. Take a tree of 1, 2, 3, 4, 5, 6, 7, 8 nodes. A tree is defined as a structure which will connect the entire network but will have only 7 arcs. This is one example of a tree on the network. The rank of the adjacency matrix is actually equal to n-1. That's what we have seen.

(Refer Slide Time: 22:48)



Another matrix which is often used in defining a project network is called a node arc incidence matrix. Let's look at a node arc incidence matrix. A node arc incidence matrix is a matrix of size n by m where n is the number of nodes and m is the number of arcs in the network and it has entries of -1 0 and 1.

(Refer Slide Time: 23:21)

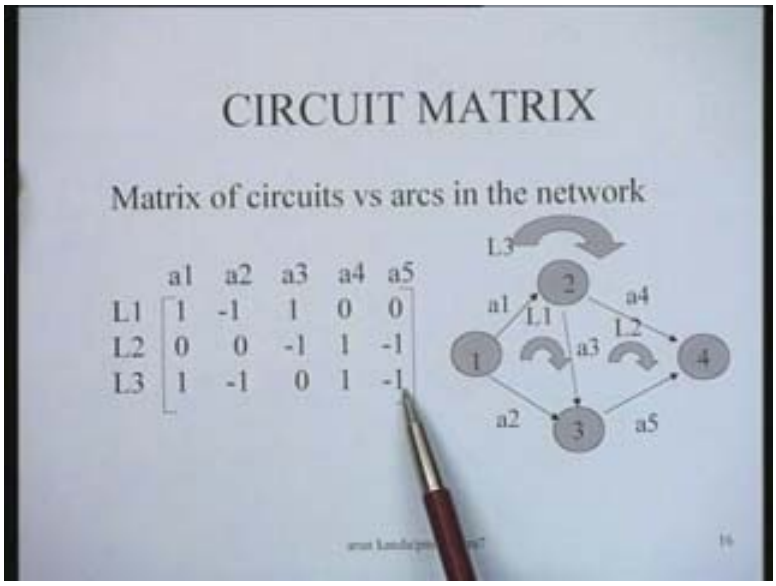


If we now take the same network for illustrative purposes we can develop the node arc incidence matrix. It has 4 nodes and 5 arcs a1, a2, a3, a4, a5. This matrix will be a 4 by 5 matrix as shown here. How it is constructed is that arc a1 starts from 1 and ends at 2. You put a -1 from where it starts and a +1 to where it finishes. Similarly arc 2 starts from 1 and ends at 3; arc 2 starts from 1 and ends at 3; so -1 and +1 here. Similarly arc a3 starts from 2 and ends at 3. In this manner the node arc incidence matrix will have exactly a -1 and a +1 in each column and nothing else. The sum of every column will be zero and a row which has all minus ones shows a source. A row which has all plus ones shows a sink. Node 1 is a source because it has all minus ones. All arcs are emanating from it and node 4 is a sink because all arcs are culminating into it. These are some of the properties of the node arc incidence matrix. In some applications the node arc incidence matrix for projects is useful especially when one is talking about project crashing, LP procedures and so on.

A third matrix associated with project networks is called a circuit matrix and it is a matrix of circuits versus the arcs in the network. What are circuits? Let's look at the same example; 1, 2, 3 and 4. In this project, find out all the loops. This is one loop. This one, a1, a3 and like this. This is another loop L2 and the whole thing is a third loop L3. If we put down L1, L2, L3 as the 3 loops here and we identify the 5 arcs a1, a2, a3, a4 and a5 we can have a matrix of this kind. For instance when you look at loop L1 you see the direction of the loop. In the loop a1 and a3 are traversed in the same direction as the loop but a2 is traversed in the opposite direction. So a1 and a3 are traversed in the same

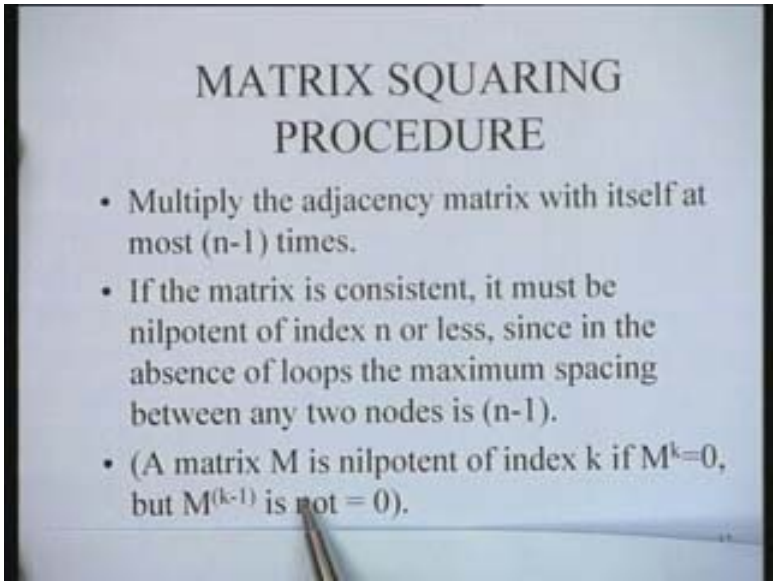
direction but a2 has a -1. So 1, -1 and 1; basically this defines the loop 1. Similarly this defines loop 2, this defines loop 3.

(Refer Slide Time: 26:21)



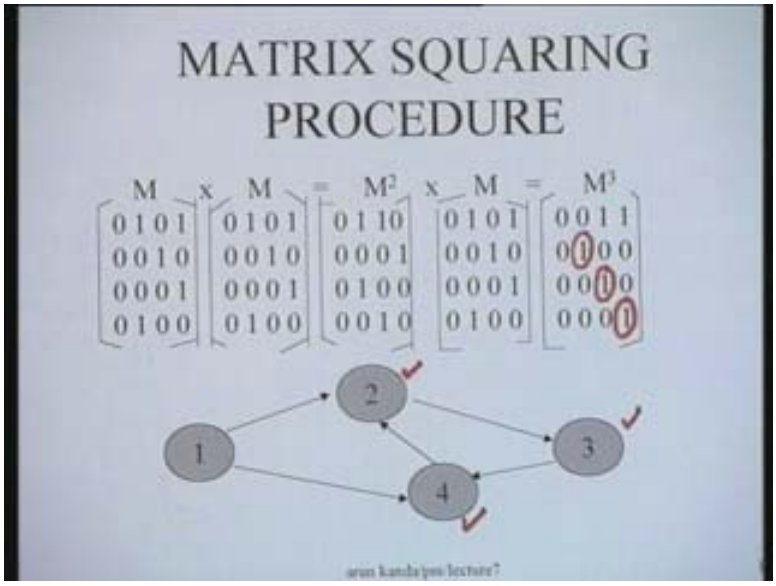
This is a matrix of again zeros, ones and minus ones which defines the structure of the circuits which are there in the matrix. This is another alternative way of representing the project network. However for purposes of checking consistency it is the adjacency matrix which is of great concern. We will be talking about procedures for looking at the adjacency matrix in some detail. The method for checking consistency of adjacency matrices is a matrix squaring procedure. The matrix squaring procedure works something like this. What it says is multiply the adjacency matrix with itself at most n-1 times. You multiply it once you get M square; you multiply it M to the power n-1 times you get M n maximum and what should happen is that if the matrix is consistent it must be nilpotent of index n or less, since in the absence of loops the maximum spacing between any two nodes is n-1. A matrix M is said to be nilpotent of index k if M to the power k is equal to zero but M to the power k-1 is not equal to zero. That's the definition of nilpotency of a matrix.

(Refer Slide Time: 27:52)



We will apply this procedure and check whether a particular matrix is consistent or it is not consistent. If the matrix is not consistent what really happens is that in the process of squaring the matrix 1 will appear on the diagonal of the matrix. That's what will happen immediately. For instance let's take an example of an inconsistent matrix for that matter. Take an example of this matrix, 1, 2 3, 4. It has a loop which is shown here. It has a loop. How this procedure works is that we develop the adjacency matrix of this particular graph. The adjacency matrix for this graph is shown here as M. It's a 4 by 4 matrix. It's like this. If we take the product of M with itself we will generate M square and M square is shown here in this particular matrix. In M square all the elements on the diagonal are still zero but if we multiply it with M again we generate this particular matrix here which is M cube and what you find in M cube is that ones have appeared on the diagonal. Here is an interesting thing which has happened. You have 1 here, 1 here and 1 here and 0 here. You see the appearance of a one in this particular situation for M is equal to M cube here shows that this is an inconsistent matrix and in fact this identifies that the second, third and the fourth node which are there on the network. Second, third and fourth nodes are involved in the circle, in this particular path. We have been able to identify the second, third and the fourth nodes. These are the nodes which are involved in the cycle. This squaring procedure automatically identifies the cycle as well and tells you that there is a problem here in the network here. The interpretation of M cube shows that from node 2 to itself there is a third order path. From node 2 itself, there is a first order, second order and third order. There is a path with 3 links. This is the interpretation of 1 which appears in this particular matrix here. By squaring the matrix in this particular procedure if you get ones it shows that there is an inconsistency.

(Refer Slide Time: 30:47)



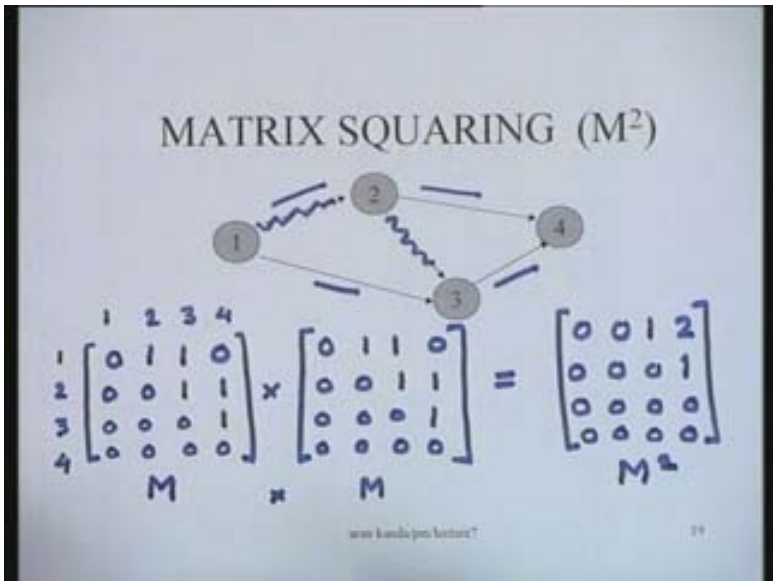
However what would happen if you have a consistent network? Let's work a small example and let's see what happens. If you get 1 in M4 on a network which has only 4 nodes actually it should be all zeros because in a consistent matrix the maximum distance between n nodes can be n-1. There can be at most n-1 arcs in a consistent network. Inconsistency would arise only when there is a loop. You would have a situation like this emerging only when there is an inconsistency. If there is an inconsistency you could still have M4 having something other than zero but if it's a consistent network M4 must be all zeros. That is the test for consistency.

Take for instance this matrix. It's a small matrix. It's a consistent matrix. In this consistent matrix let us try to identify what is going to be the adjacency matrix for this particular graph? From 1 there is no arc to 1 itself. There is an arc to 2 and there is an arc to 3 and there is no arc to 4. That's the first row. This is node 1, this is node 2, this is node 3 and this is node 4 and we are going to node 1, to node 2, to node 3 and to node 4. From 2 there are two arcs to 3 and 4. There is an arc to 3 and there is an arc to 4 and these are the two zeros and from 3 there is only 1 arc to 4 and all others are zeros and from 4 there is no arc. This is basically all zeros. This is the adjacency matrix for this particular graph. This is M. Suppose I was to multiply this matrix with itself what would you be getting here? That is M into M would give us actually M square. What is that value of M square? Let us find out what that value of M square is? The first element here you would be multiplying the first row with the first column. The first column is zero. This will be all zeros. The first row with the second column will actually generate all zeros. The first row with the third column will actually generate 1. The first row with the fourth column what is it going to generate? It is going to generate 1+1 which is 2. Let's look at the second row with the first column. It's going to be zero. The second row with the second column it's going to be zero. The second row with the third column is zero. The second row with the fourth column will be 1. Let's look at the third row. Third row with first column will generate zeros. The third row with the second column will generate zeros.

The third row with the third column will be all zeros and the third row with the fourth column will be all zeros and let's talk about the fourth row. Because it is zeros everything will become zeros here. So M square value is actually this. Let's talk about the interpretation of these numbers because that's important for us.

M square shows actually second order links. It shows that from node 1 to node 3 there is one second order link. From node 1 to node 3 which is the second order link? The second order link is this and this. (1 2) (2 3) is a second order link from node 1 to 3 and that is actually being shown in this matrix M square. It shows that there are 2 second order links between node 1 to node 4. From node 1 to node 4 there are 2 second order links. Which ones are these? 2 second order links means 1 2 4 is one such path. This one and 13 4 is another path. 1 2 4 and 1 3 4; these are two paths and therefore they are second order. They both are involving 2 arcs. The interpretation of M square is the number of second order links that exist between node i and j in this particular network. This is in fact M square.

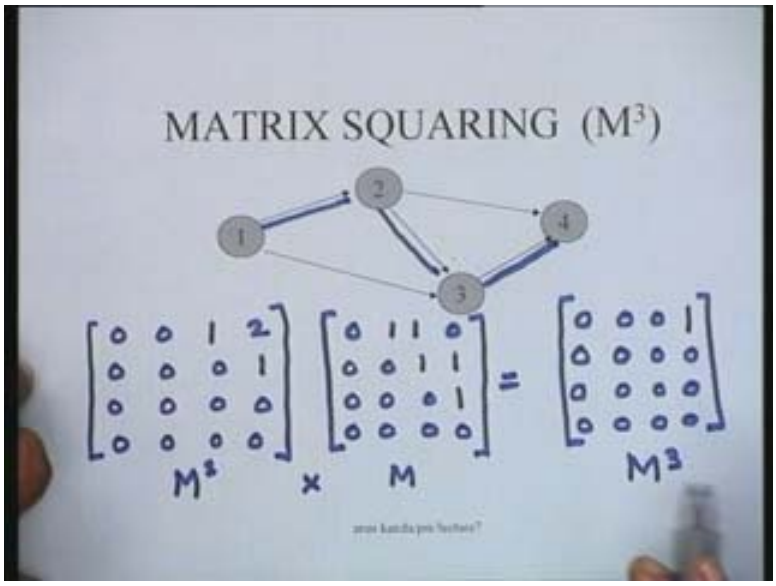
(Refer Slide Time: 36:27)



Let us utilize this information to generate the value. Let's generate M cube for this and for instance we found that M square for this example is 0 0 1 2, 0 0 0 1, 0 0 0 0 and 0 0 0 0. This was actually nothing but M square. Let's multiply this with M. The value of M is 0 1 1 0, 0 0 1 1, 0 0 0 1 and 0 0 0 0. This was the value of M. On multiplication I would be getting the value of M cube. Let us try to generate the values of M cube. You can see here first row into first column will be zero. First row into second column will be zero. First row into third column will be zero. First row into fourth column will be 1. Second row into first column is zero. Second row into second column is zero. Second row into third column is zero. Second row into fourth column is also zero. Third row will be all zeros and the fourth row will also be all zeros. When you discover the matrix M cube it shows that there is 1 third order link from node 1 to node 4. Which is that third order

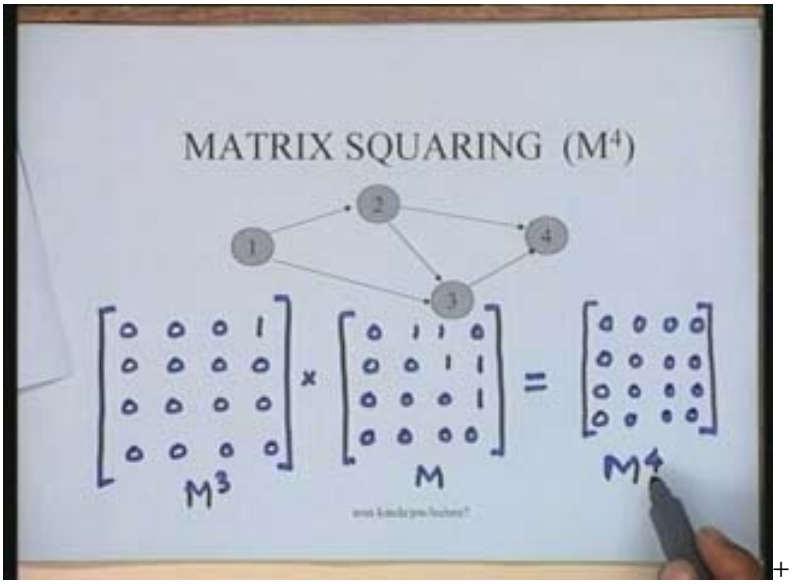
link? It is this one; this, this and this. It involves 3 arcs. It's a third order link from this to this.

(Refer Slide Time: 38:36)



Let us complete this procedure by calculating M to the power 4 and try to find out whether the matrix is consistent or not. What we can do is we can generate the value for M4 simply by multiplying the matrix and cube with M itself. M cube we had just determined just now. M cube is this particular matrix. This is the matrix M cube. I multiply this with the matrix M. The matrix M you will recall is nothing but 0 1 1 0, 0 0 1 1, 0 0 0 1 and 0 0 0 0. This is nothing but the matrix M and by this product I would get M to the power 4 and in this case what do I get? You find that this matrix is actually nilpotent of index 4. This matrix is nilpotent of index 4.

(Refer Slide Time: 40:05)



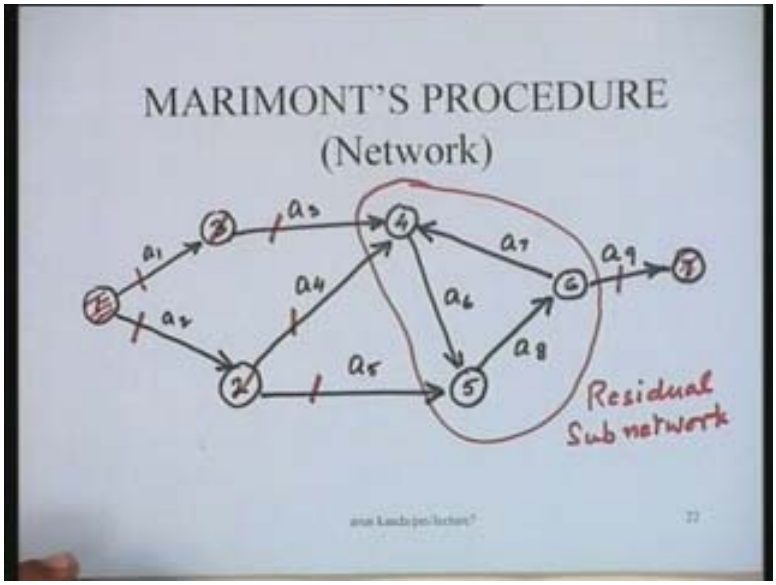
What does it show? It shows that the matrix is consistent. That is if there are 4 nodes then M4 must necessarily be all zero. There should be no fourth order parts in this particular network. M cube can exist. If a matrix is nilpotent of order M to the power n or less it is consistent. If by multiplying up to M n you still find that there are ones here or somewhere it shows that there is an inconsistency and the appearance of ones on the diagonal at any stage shows you or reveals the various kinds of paths which exist at that particular point in time. This was the matrix squaring procedure which can be utilized for determining the consistency of project networks.

Finally let's look at Marimont's procedure. In fact Marimont's procedure is best illustrated on a network. We can take any network and try to illustrate Marimont's procedure on the network itself. Let us say for instance that we have a project network which is nodes. Let's take a network of this kind and try to find out whether it is consistent or not consistent by applying Marimont's procedure. In the procedure of Marimont's it is not necessary that the nodes be numbered. We can number them. Let's suppose we number these nodes arbitrarily 1, 2, 3, 4, 5, 6 and 7. Let's suppose that there are 7 nodes. In this case the number of arcs is a1, a2, a3, a4, a5, a6, a7, a8 and a9. There are 9 arcs and 7 nodes in this network and you want to apply Marimont's procedure. By looking at the network you identify the sources and the sinks.

It's a very simple procedure. Node 1 is actually a source and node 7 is a sink. This is a source and this is a sink. You delete the arcs which are emanating from the sources and you sort of delete this particular node as well. This is disconnected from the network and you similarly try to disconnect node 7 and the arc which is connecting it. We delete this. In this process what happens? These two becomes sources again; node 2 and node 3 have become sources again. What does that show? It shows that we can continue this procedure. We can then delete these arcs which go from these sources. These arcs which go from these sources along with nodes 2 and 3 and similarly here when we delete this

sink this does not become a sink or it does not become a source. At this stage are we left with any other sources and sinks? This arc is deleted, this arc is deleted. But this node still has an arc. This particular node still has an arc coming in and this node has. We have been able to delete nodes 1 2 3 and these arcs and this node and what is left, that we are not able to delete; this particular portion of the network at all. We are not able to delete either node 4 or 5 or 6 whereas we have been able to delete the other nodes 1 2 3 and 7. This is what we call a residual sub network. The presence of the residual sub network actually shows that there is an inconsistency and actually you will find that in this particular example there is a cycle present in these particular nodes. That means in the Marimont's procedure if you had been able to continue this procedure right through, you would not be left with any residual network. You would have been able to delete everything. That is what it would have been possible to do.

(Refer Slide Time: 45:44)

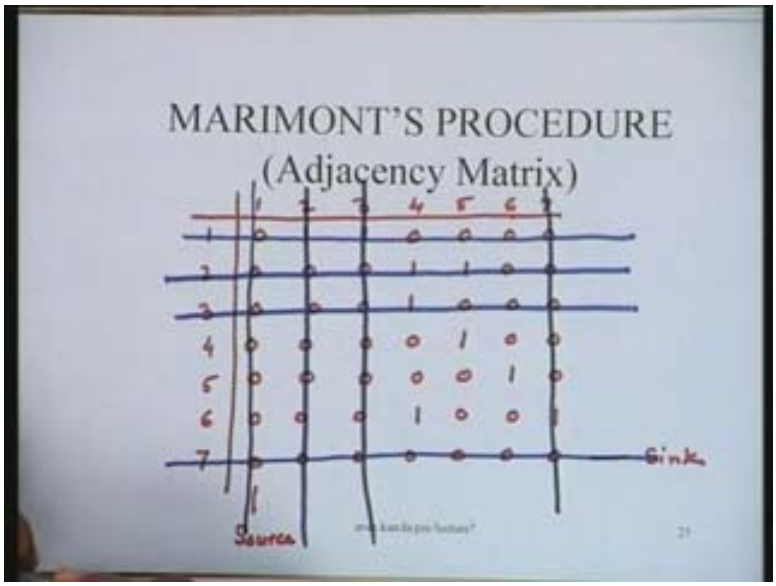


This very procedure could be applied on the adjacency matrix also rather than on the graph. How can it be applied on the adjacency matrix? I will just give you a hint. If you take the same example the network that we considered had 7 nodes. If we construct a 7 by 7 matrix you find that in the original network we had from 1 there was an arc to 2 and there was an arc to 3. Similarly from node 2 there were 2 arcs to 4 and 5 and there were no other arcs. From node 3 there was only 1 arc to 4 and there is no other arc here. From node 4 there was 1 arc to 5. From node 5 there was 1 arc to 6. That was all. From node 6 there were 2 arcs to 4 and 7 and from node 7 there was no arc. If you take a vacant row this shows that the node 7 is a sink. A vacant column that is node 1 is a source. Look at the adjacency matrix and identify the sources and the sinks and then you are supposed to delete the arcs which are emanating from the sources and the sinks. How do you do that?

For instance node 1 is a source. What I do is I delete column 1 and the corresponding row 1 and the corresponding row 1 will automatically delete these two arcs which was a1 and a2 in the original network. I am left with a 6 by 6 sub matrix after deleting node 1.

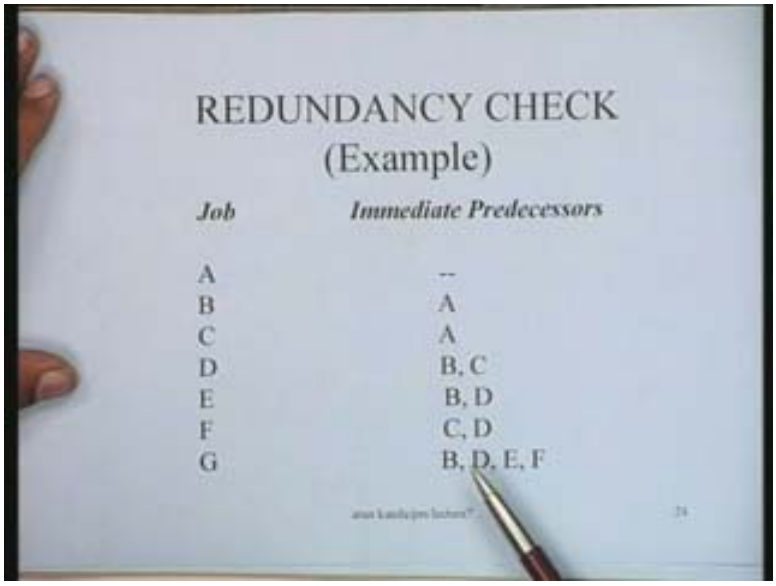
Similarly if I try to delete node 7, I would delete node 7 and I would also delete the row and column corresponding to that particular node. After deleting nodes 1 and 7 I would have this particular representation. In this remaining representation I try to find if there are any sources. You find that 2 and 3 are in fact the sources which are left here. I delete 2 and the corresponding column 2 and similarly I delete node 3 and the corresponding column 3 in this process. I have been able to delete nodes 1 2 3 and 7, the corresponding rows and columns and I am left with this residual sub matrix. This sub matrix does not contain any sources. This is a residual sub matrix. It shows an inconsistency.

(Refer Slide Time: 49:32)



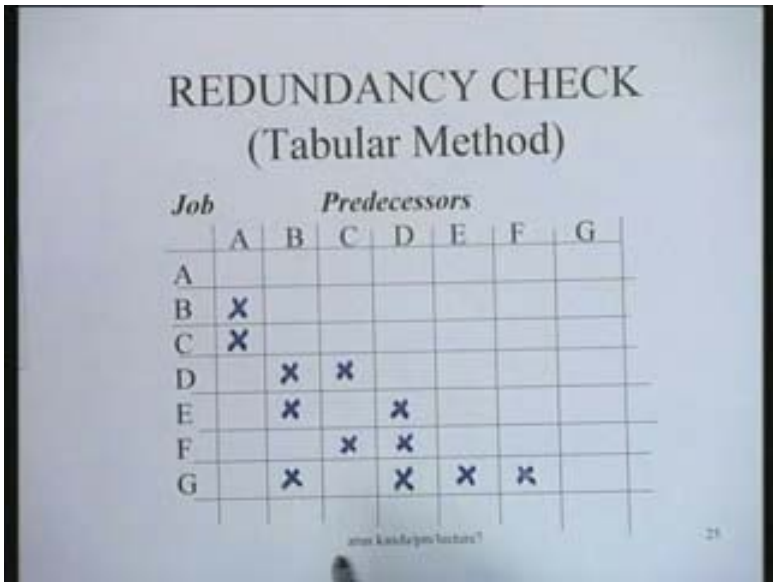
The Marimont's procedure can be applied on the adjacency matrix as well in the same manner that it can be applied for the whole thing here. So far we had been talking about consistency checks. Let us now talk about redundancy. What is redundancy? Redundancy is the presence of unnecessary information or additional information in the predecessor set. Let's take an example of jobs like this A to G and with the corresponding predecessors as they are shown here.

(Refer Slide Time: 50:12)



Once these jobs are shown here we would like to check whether there is any redundancy in this set of jobs. A simple redundancy check can be carried out on a table and the tabular procedure is something like this. The information that was given about the network A has no predecessors B was shown to have a predecessor A. C was shown to have a predecessor A, D has predecessors B and C and E had predecessors B and D. F had predecessors C and D and G had predecessors B, D, E and F.

(Refer Slide Time: 51:00)

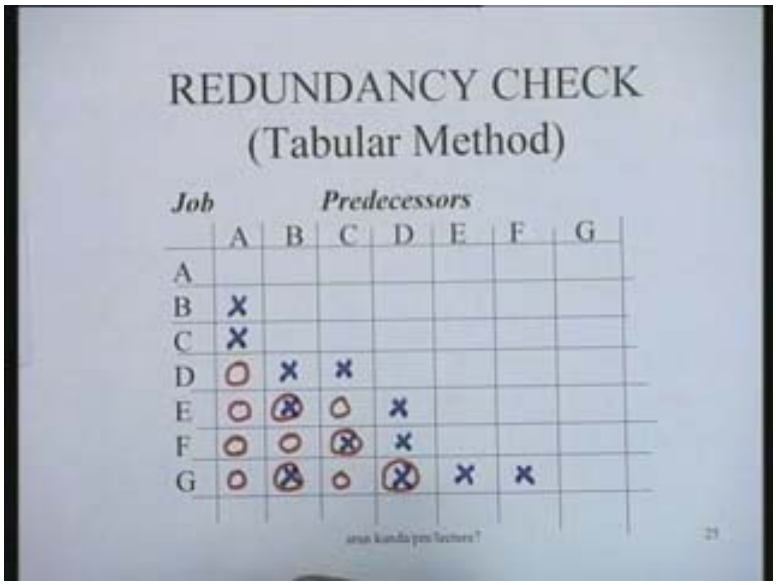


This is the information; the same information which is given to us which you would like to check for redundancy. Mind you this list must be in topological order before you start

implementing the procedure. This list is in topological order in this particular situation. Then what you do is B has nothing. C has only A and it is taken care of. Look at D. D has B and C. So look up B and C. We can use a different colour here may be. We can identify. B has A as a predecessor. This circle shows that A is an implied predecessor of D; similarly C has A. This is A. Look at E. E has B and D. Just transfer this information down and for D all the three; so you transfer this information down for all the three and this is already there and then you go down to F. When you go down to F you find C and D. Look up the rows corresponding to C and D and you find that these three become the implied predecessors of F. Then you look up these, B D E and F. For B there is only one. For D there are these three, all these three. For E there are these four and for F there are these four which is already taken down.

This is a procedure, where corresponding to each of these crosses you check up that particular job and transfer its predecessors down below. That's all. For G if I look up for E, these four. So I transfer the four downwards. It's like trying to say that these are now the distant predecessors of this particular job or the implied predecessors. In this process you find that we have crossed. These 4 crosses have been circled. This shows that there are four redundancies in this particular example. We have been able to identify the redundancies.

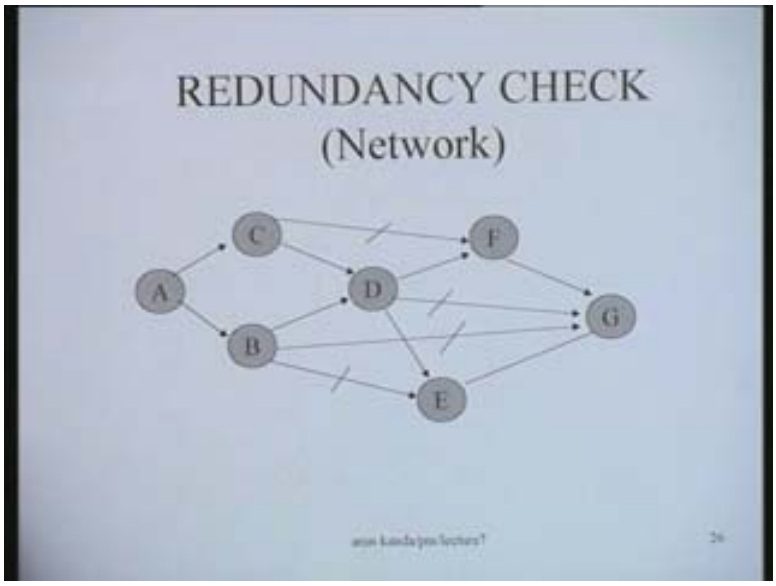
(Refer Slide Time: 53:12)



These are the four redundancies which are crosses and which are circled. On a network the same information, the four redundancies which are crossed out are shown here. These are the ones which are crossed out. That means these are the unnecessary arcs which you don't have to specify. This has been discovered through redundancy. For F we have shown C as a predecessor. It need not be shown because D is a predecessor of F and C is an implied predecessor. So we don't have to say that C is a predecessor of F. This is a redundancy. In some cases redundancies might be easy to spot. Otherwise you will have

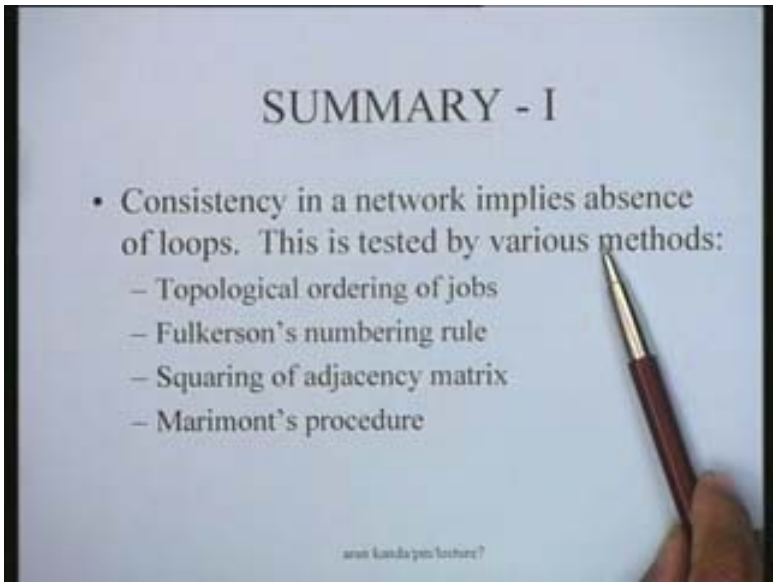
to look for a systematic procedure by which it can be done in this particular manner. This is a procedure that you can adopt for removing redundancies.

(Refer Slide Time: 54:08)



Let us see what have tried to do in this particular lecture. By and large we have looked at the problem of consistency in a network and we have seen that consistency in a network implies the absence of loops and this can be tested by a variety of methods.

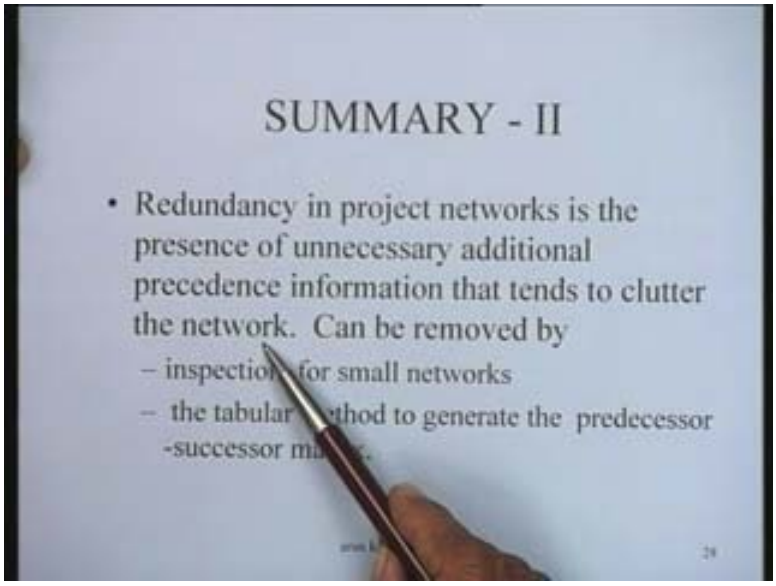
(Refer Slide Time: 54:25)



The four methods that we adopted or which we have studied in today's lecture are topological ordering of jobs, Fulkerson's numbering rule, squaring of the adjacency

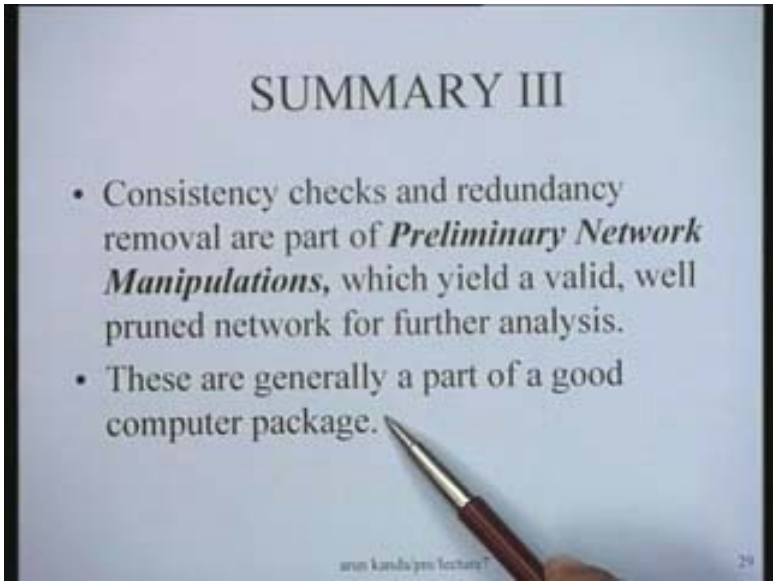
matrix and the Marimont's procedure for reducing this. Then we have looked at procedures for redundancy and we have seen that redundancy in project networks is the presence of unnecessary additional precedence information that tends to clutter the network.

(Refer Slide Time: 54:58)



These can be removed by either inspection for small networks or by the tabular method to generate the predecessor successor matrix which is what we had done for this example and finally we have seen that consistency checks and redundancy removal are basically a part of what we call preliminary network manipulations which yield a valid, well pruned network for further analysis and these are generally a part of a good computer package.

(Refer Slide Time: 55:31)



They would be necessary for developing a good project network for subsequent analysis. Thank you!