Linear Programming and its Extensions

Prof. Prabha Sharma

Department of Mathematics and Statistics

Indian Institute of Technology, Kanpur

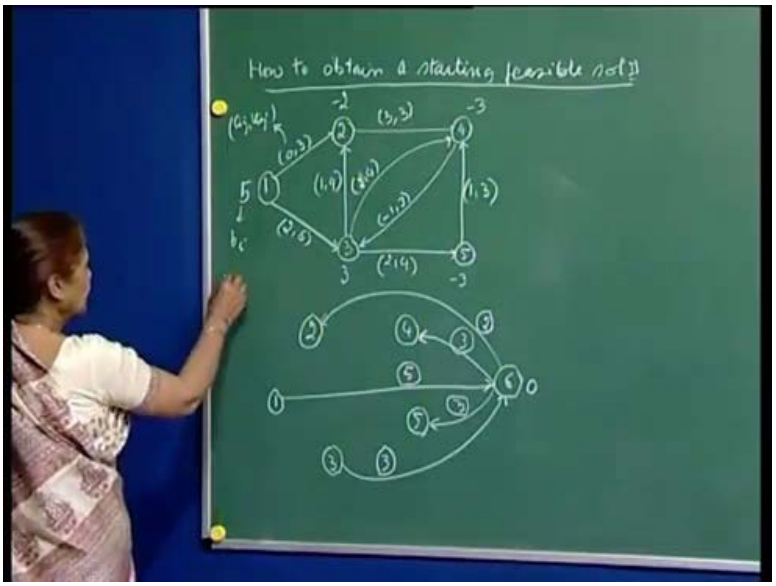
Module No. # 01

Lecture No. # 30

Starting Feasible Solution Lexicographic Method for Preventing Cycling Strongly feasible solution

So, let us today first discuss the issue of obtaining a starting feasible spanning tree solution, to your network flow problem; is min-cost flow problem. So, here is the problem with 5 nodes and these numbers indicate your, yeah, I should have set somewhere here, these are your b i's. So, node 1 and node 3 are the two source points and nodes 2, 4 and 5 have demands.

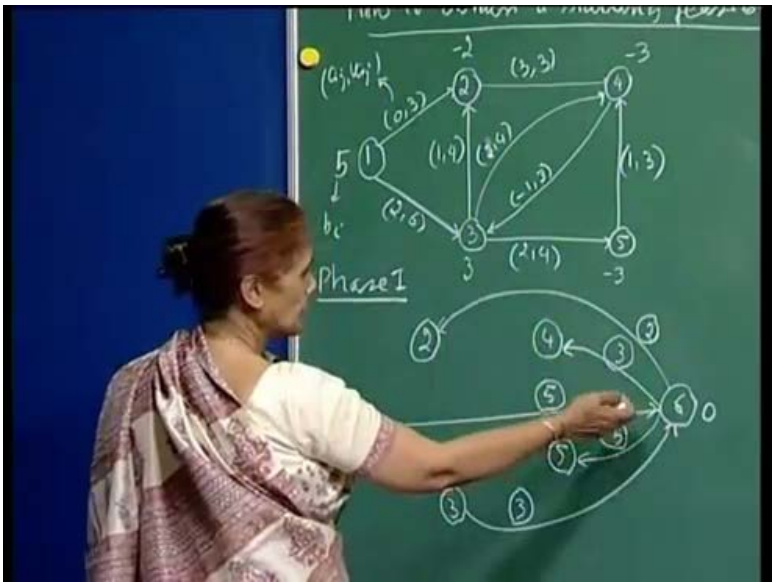
(Refer Slide Time: 00:33)



So, the demands are shown with the negative a i's. Then, these numbers is, first number is the cost and the second number is the upper bound on the arc on the flow. So, there is more than one way always of obtaining a basic feasible solution, starting feasible

solution. I will indicate both to you, but we will discuss this one in the term more detail and, the, this is phase I; so, that means phase I.

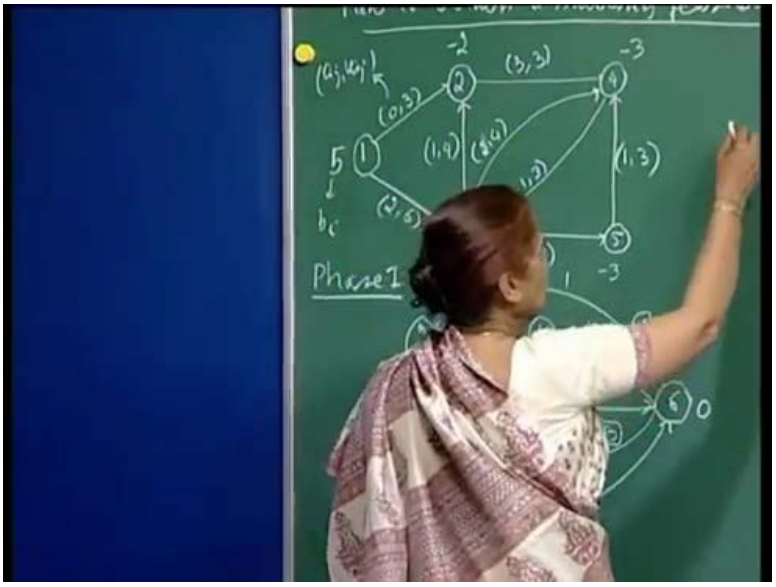
(Refer Slide Time: 01:07)



So, the idea is that you add artificial, artificial, node to the network and then connect all nodes which have positive supplies to with the arc coming into 6 and all nodes which are your markets, that is, 4 and 5 you connect the, if the arc has to go from 6 2 that node.

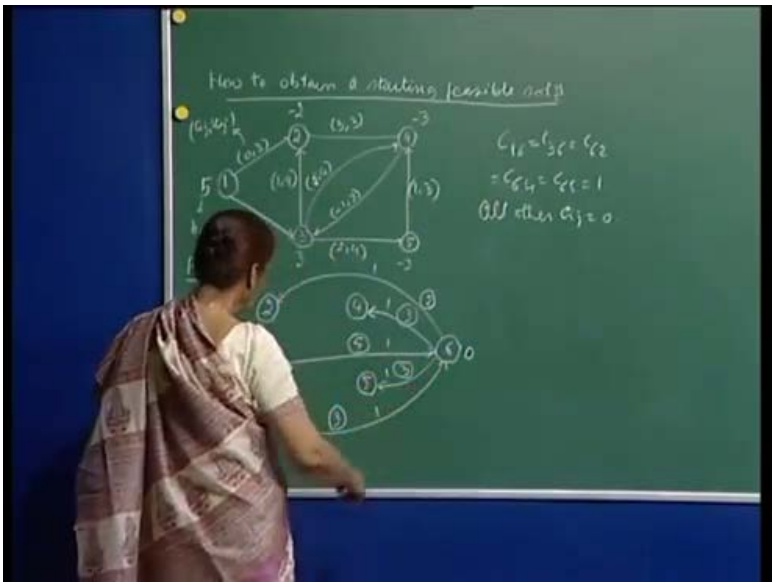
So, you can see that and of course, in the supply here would be 0. Therefore, you see, this point will then get 8 units, because 5 plus 3 either total supplies available in the network. So, 8 units will go to 6 and then these 8 units will get 3 distributed to 2, 4 and 5. So, 2 has demand of 2; 4 and 5 both have demands of 3 units. So, this is the thing. So, you start with this and you can, this just like your phase 1 that you sub you start your basic feasible solution with the artificial arcs.

(Refer Slide Time: 02:19)



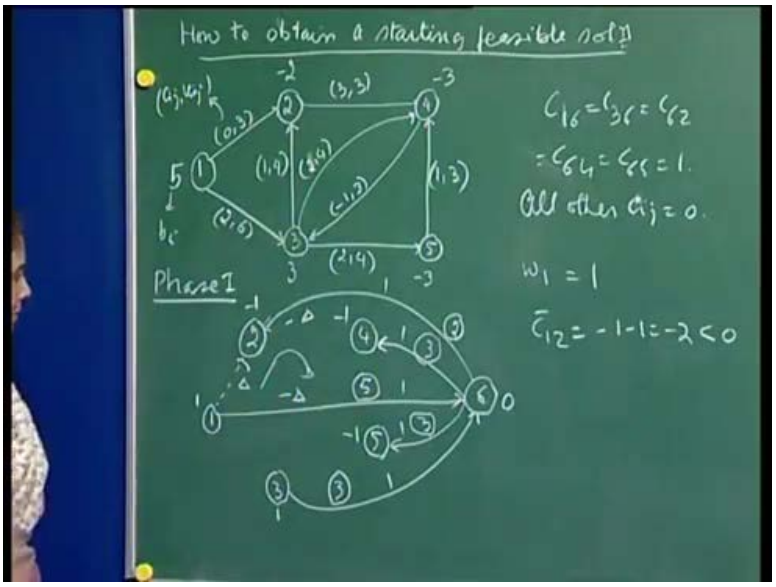
So, here your starting solution 1 6 x 1 6 is 5 and of course, the cost to these arcs will be 1 just like in phase 1 and the original cost will be all 0. So, what we are saying is that you are in phase 1.

(Refer Slide Time: 02:31)



Your c, all I will write out this here c 16 is equal to c 36 is equal to c 62 equal to c 64 equal to c 6 5 is 1. All other c i j, c i j, equal to 0; that means the original arcs cost 0 exactly phase 1. So, you just see the correspondence at every step. So, here, this is your this thing, and now, we will compute the dual solution. So, quickly you see you can immediately see that for this arc for example, for a supply point.

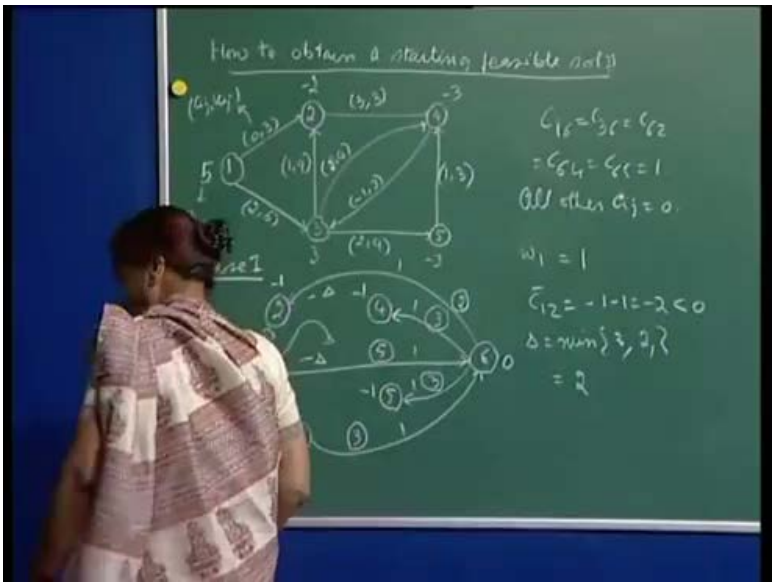
(Refer Slide Time: 03:19)



It will be w 1 minus w 6 which is 0; w 1 minus w 6 which should be equal to the cost here is 1. So, for all the supply points, the dual cost would be 1, and for all these things, for all demand points, it will be minus 1 exactly the same, and so, we begin our simplex, network simplex algorithm. So, you can see that start with the node and I will use the first encountered arc, the first beneficial arc encountered as the candidate for coming into the basis. So, if you look at arc one 2, what is the cost, the relative cost? The relative cost c 12 bar is see the arc is 1 2 and the 0, the cost is 0 here. So, c 1 2 is 0. So, this will be minus 1, and then, w 2 which is this minus 2, this is less than 0. So, in fact, I do not need to repeat these steps but just may be 1 or 2 iterations I will show you. Therefore, this is an arc for coming into the basis, and so, you have to add this arc.

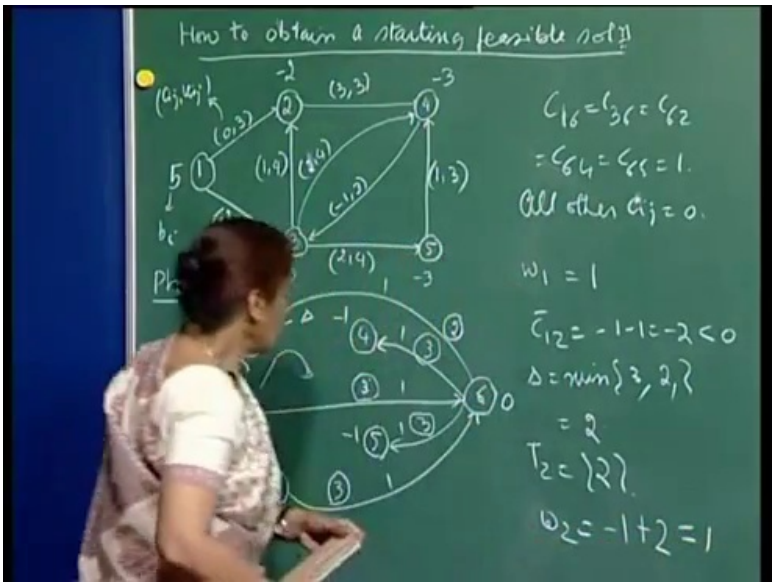
The moment you add this arc to your tree, it becomes does a cycle and you want to increase the flow. So, the orientation of the cycle is this. So, if this is plus delta, this will be minus delta because the arc in the opposite direction and here also it will be minus delta. So, we just look at the upper bounds here. See for example, delta cannot be more than 3.

(Refer Slide Time: 04:55)



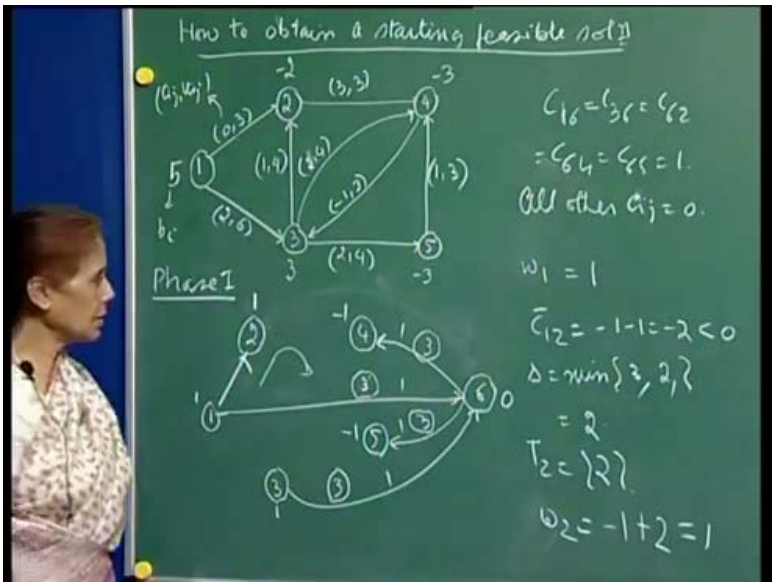
So, your choice of delta has to be minimum of 3. Then here to the flow on this arc is 2. So, you cannot reduce the flow by more than 2. Therefore, this will be 2 and here of course, this is 5. So, that is fine. You have this for the system; so, that means the incoming want to comes at a level 2, and in place r 2 6, leaves the basis. So, we will do the calculation here and I will make this arc basic. So, this will become the updated; this is 3. Before I enter this, remember we have to then update the dual solution and what the rule the shortcut we were using was that since arc 6 2 is going to leave the basis.

(Refer Slide Time: 05:23)



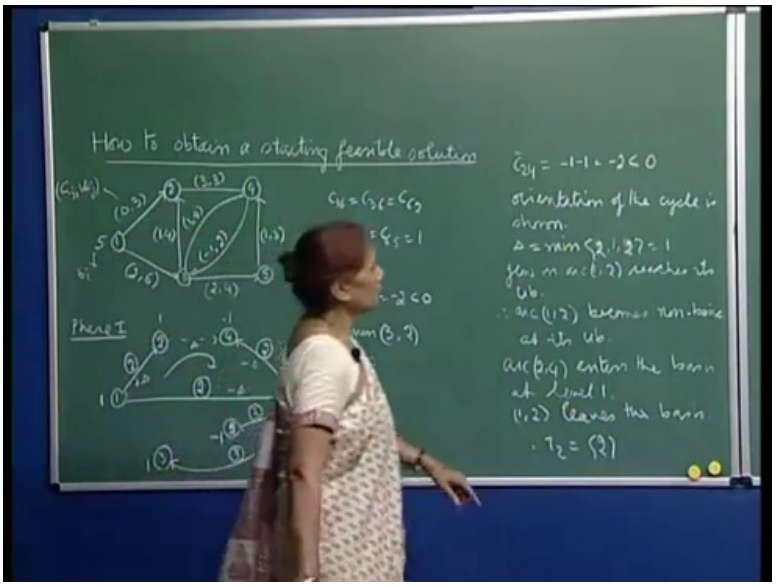
So, if we drop this from your original tree, then you see what are you left with? You are left with this one. So, t is therefore, and, and, your root node is 6 which belongs to t 1. So, t 2 just consists of node 2, and since c 12 bar is minus 2, the dual value here would go up by, would get reduce by minus 2, which means your nu, your nu, w 2 bar, your nu w 2 is old minus plus 2 minus of minus 2 which is 1.

(Refer Slide Time: 06:24)



So, therefore, the iteration is very quick; this is simply 1. This will remain 1, well, not now, sorry, it will remain 1 because this was part of t 1. So, this is this and this arcs, so, this arc goes out of the basis; so, this is your new tree. So, one artificial arc has left to basis. So, let us compute the, the, relative prices again.

(Refer Slide Time: 06:56)



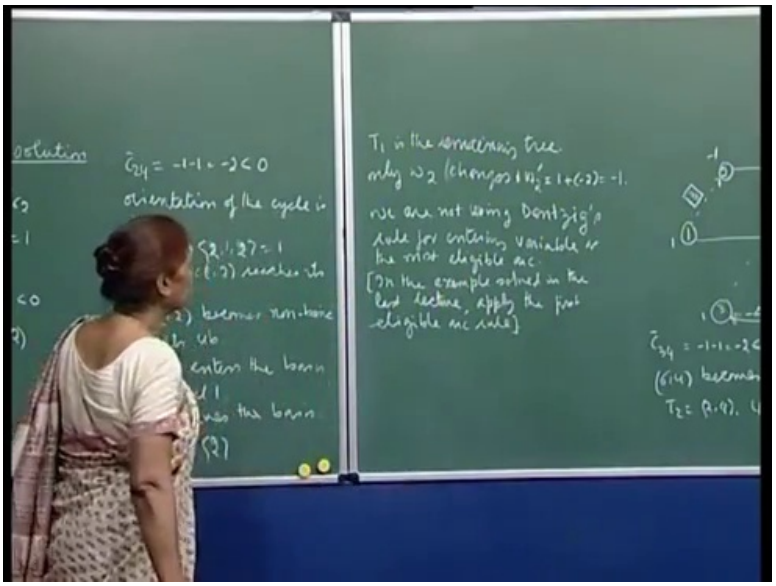
So, c 2 4 bar is minus 1 minus 1; so, that is less than 0 minus 2 is less than 0. So, we are, I will say that this arc, if it enters the basis, they will be in improvement in the cost and I am taking the first arc that I get which is eligible, and so, the orientation of the cycle you see 2 4 has to enter, then the, and since the flow has to increase on 2 4, the orientation of the cycle will be like this. So, it will be plus delta here minus delta minus delta and plus delta, and so, yeah, I am that, that, yeah, right, fine.

So, then delta comes out to be minimum of 3 1 3, because you already have 2 units of flow here and the flow at most can be a 3 units on arc 1 2. So, therefore, delta will be minimum of 3 1 3 you can see, because actually this is only 2 2. So, I should not have written 3 3, it should be 2 2. In anyway, the minimum value is not affected. So, this is delta equal to 1, and so, flow on arc 1 2 reaches its upper bound. So, once I increase the flow by one unit here, then the flow has gone up to 3 units, and so, this reaches its upper bound. So, since I have to drop one arc and none of these arcs become, the flow does not become 0.

So, they will remain basic; 2 four is becoming basic. So, I will drop 1 2 from the basis; that means I will make the status of arc 1 2 will change from basic to non-basic at its upper bound. Therefore, 1 2 leaves the basis and 2 4 enters. So, when 1 2 leaves the basis in the old tree, you see if this leaves the basis, then your t 1 is all this and t 2 is only containing arc node 2, and therefore, t 2 is 2, and so, your add up and since the arc 2 4 is

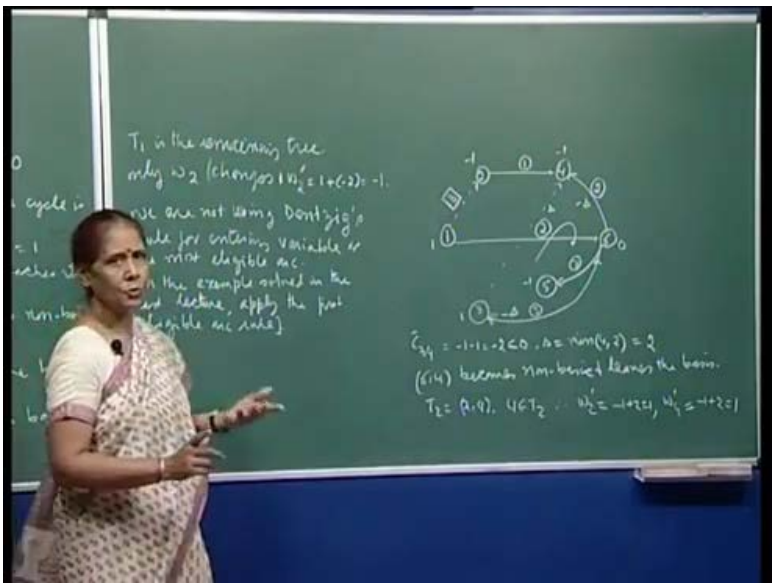
entering; that means p q, so, p belongs to t 2; p 2 p belongs to t 2. Therefore, you will increase the w 2 prime by the c 2 four bar.

(Refer Slide Time: 09:04)



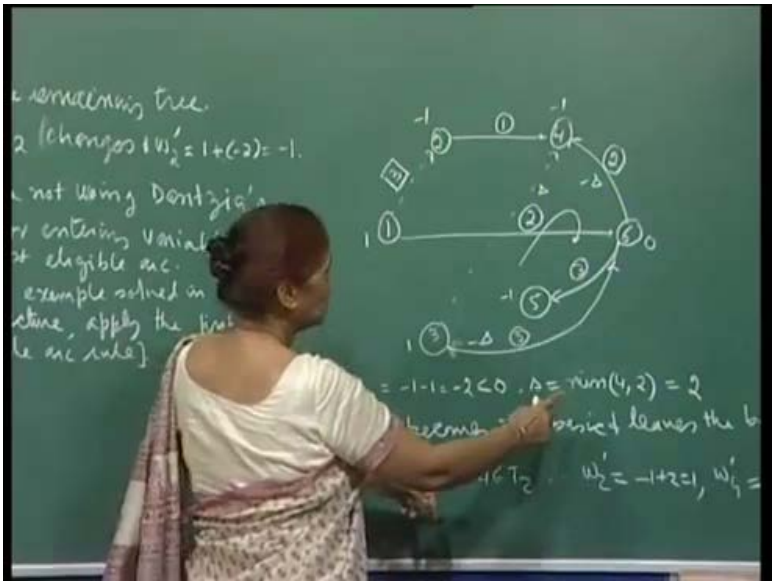
So, therefore, 1 plus minus 2 because c 2 four bar is minus 2. So, for, if, when p is in t 2, then you add the c 2 four the, the, relative price of the incoming arc, and if q belong, then you subtracted. So, it is 1 plus of minus 2 which is minus 1. So, that is the only change in your this thing and of course you update the flow.

(Refer Slide Time: 09:27)



So, I have shown it, yeah, so, therefore, the flow, the new flow I am showing you here. So, the new basic feasible solution, this is non-basic and this is the current flow. I can something, I think originally the flow was 3 here on these 2 arcs, and so, I have probably updated the flow right here also, but in any case, this is the current basic feasible solution. So, it is 2 2 and this flow is become 1. The arc 2 4 enters the basis at level 1. This becomes non-basic and I have shown you, fine. So, now, we continue with the, so, as I said that I am not using the Dantzig rule for entering variable for, you know, where you choose the most eligible arc.

(Refer Slide Time: 10:10)

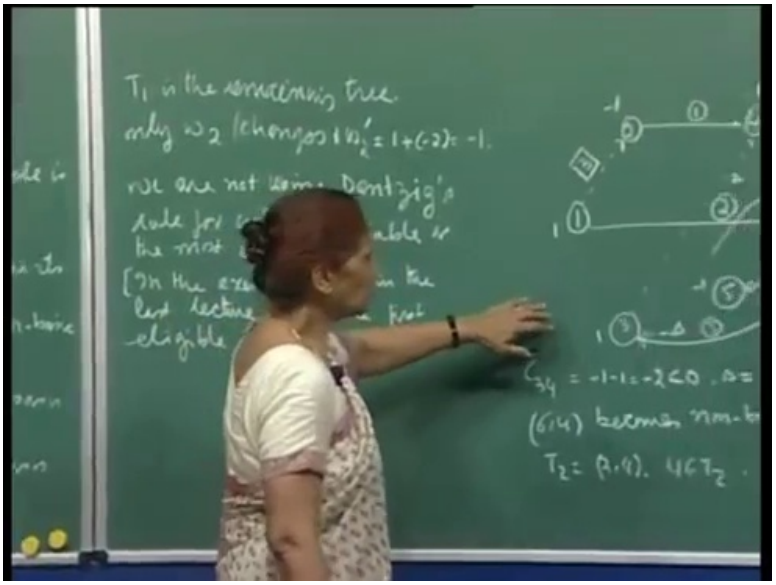


I simply choose the first incoming and of course, you might say what would be the weight of first incoming, and of course, that of course you may decide whichever way. So, the first arc that you hit as a eligible arc, you can enter it into the basis. So, in the example that I solved in the last lecture, the min-cost flow problem, I would like you to up with there I use the Dantzig rule for entering a arc in the basis. Now, try to apply the first eligible arc rule and see if there is a difference any number of iteration, of course one example is not enough but fine, you might just check for that. Now, I continue with the one more iteration of phase I for min-cost flow problem.

So, now, c bar 3 4 and of course, you may have notice by now that arcs for which the tail has value 1 and head has value minus 1. Those are the ones for which the relative price will be minus 2. You might have discover that by now. So, c bar 3 4 is equal to minus 2 is less than 0, and so, here, again when you give the orientation of the cycle, because flow on 3 4 has to go up, so, this is plus delta minus delta and this is minus delta.

So, here, delta will be 2 and here 3 4 the flow can be at most four units. So, therefore, minimum of 4 and 2 which is 2. Therefore, the flow can increase by 2 units on the arc 3 4 and that case 6 4 will become non-basic, and so, here also you can immediately see that when 6 4 is leaving the basis, then your t 2 is 2 4; t 1 is this one, 1, 6 and 3. So, 2, 4 and then 4 belongs to your leaving arc is, incoming arc is 3 four. So, q, q, is 4 and 4 belongs to t 2.

(Refer Slide Time: 12:07)



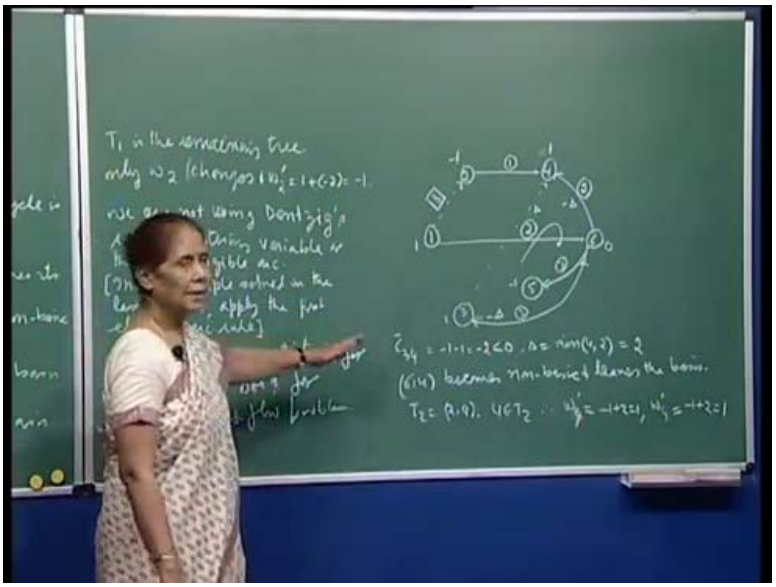
So, your, I should I write this, this should be w 4 prime. So, w 4 prime which was minus 1, so, minus 1 and then this is minus 2, so, plus 2 which becomes 1. So, w 4 primes plus 1. So, I should update this 2 1, and then, the flow has to be updated; that means the flow here will become 2; this will become non-basic; flow here will become 1. So, you will continue and you see that now you have got read of one more artificial arc.

So, the idea is that you continue with these phase I till you have driven out all the artificial variables from the artificial arcs from the basis and you have a flow vector

consisting of the original arcs. In case you have, you still have, see artificial arc at 0 level, then you will again no methods how to enter any of the original arcs in the basis to form a cycle which, which, can form a cycle with the current consisting, a current tree consisting of the cycle must contain that artificial arc at 0 level, you do that, and then, you can enter, because they will be no change in the flow since the outgoing arc has 0 flow. So, your delta will come out to be 0.

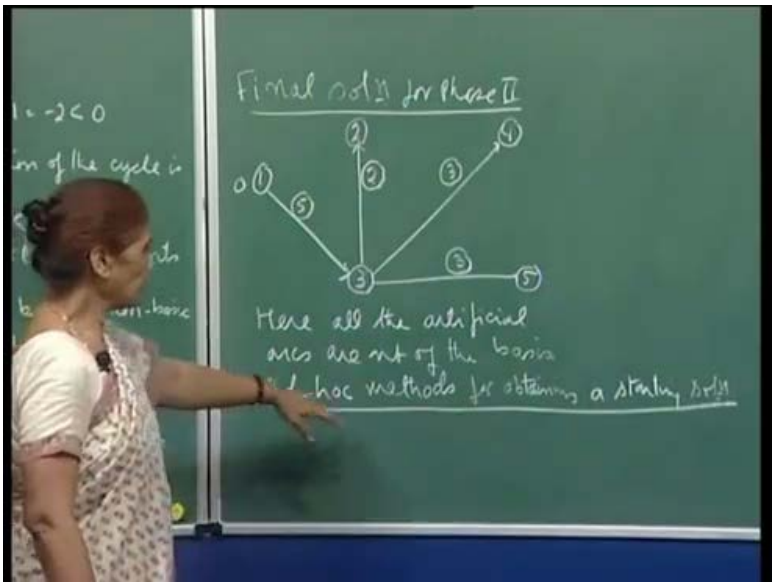
So, you know, you can handle that part how to, finally, obtain a basis now of course, the issue, that is, left is I have not yet discuss that and I plan to do that as I promised for bounded variable transportation problem also and the same thing almost applies here. As I told you, the only difference between a min-cost flow problem and a bounded transportation problem is that the for the min-cost flow problem, it is a general network; for bounded transportation, it is bipartite.

(Refer Slide Time: 14:03)



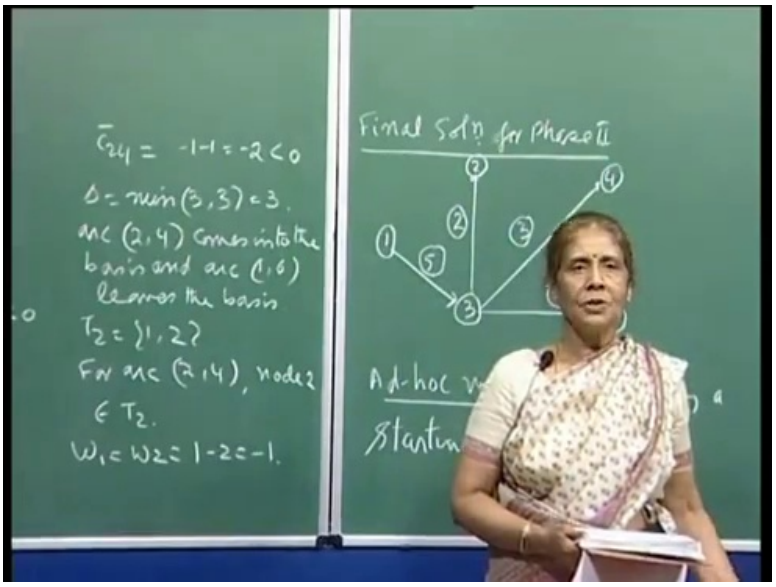
So, I would like to give you sufficient condition sufficient condition for a feasible solution, feasible solution, for min-cost flow problem. So, this we will discuss once I have talked about the max-flow problem, but now, here, finally I just want to show you what will be the final solution that you will obtain at the end of phase one and then you can begin your phase I .

(Refer Slide Time: 14:39)



So we continue with phase I and I want just to show you this is what you will get at the end of phase I. So, all artificial variables are out of the basis. I have written here final solution of phase II. What I meant is that final solution of phase I, and now, you can begin your phase II. So, phase II you will put the 0 thing here, or so, the root node is this. So, you will put your w 1 as 0, and then, you can immediately compute your 2 3 4 5. The w is corresponding to the other nodes and you can begin your network simplex algorithm. So, no problem; so, as I was telling you that it is possible that your phase I may end with some artificial variables in the basis at 0 level, then you know how to drive them out and get flow vector consisting of the original set of arcs, and as I said there is also important that you should know, you should have sufficient conditions for the solution being that, for the problem being feasible, and I would like to discuss that in one go when we are we have talked of the a max-flow problem. Now, ad- hoc methods for obtaining a starting solution.

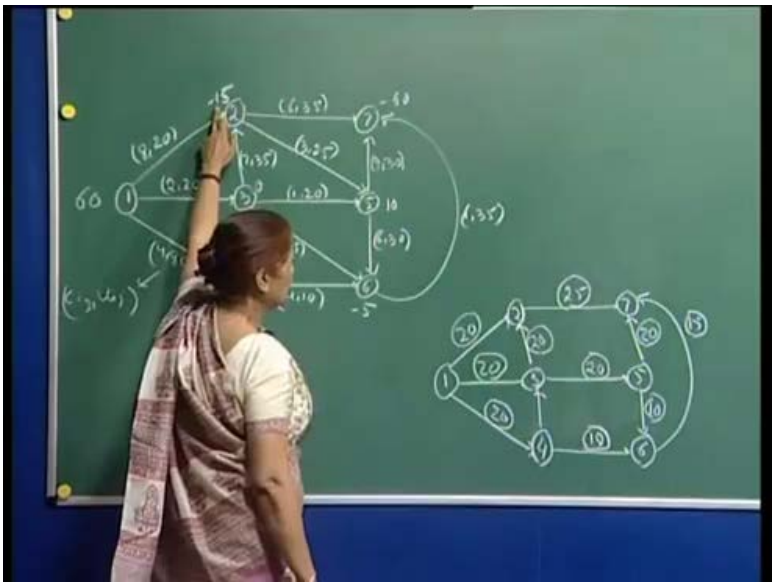
(Refer Slide Time: 15:48)



The idea simply is that you just start from your first node whichever is the supply point send as much flow as you can along the arcs which are going out of it. Then you go to the next node, which are the surplus node. You again from that send it keeping in mind the upper bound constraints.

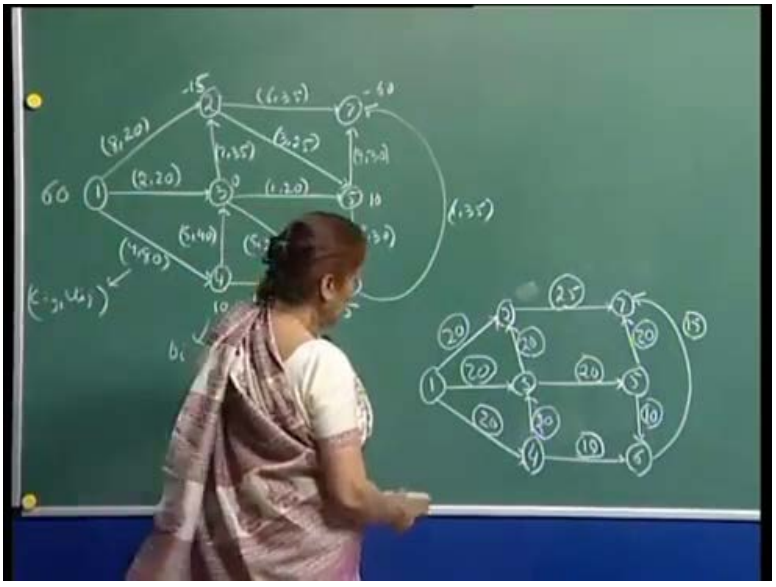
So, you just like that you can try to satisfy all the demand constraints and use up all the supplies and then you can reduce the solution that you obtain. That will not necessarily be basic feasible solution; that means it will not being a spanning tree, it may have more than m minus 1, arcs m n minus 1. If the number of nodes is n, it may have more than n minus 1 arcs. So then, I will give you a method to reduce the given feasible solution to a basic feasible solution.

(Refer Slide Time: 16:37)



So, the ad-hocs method I will demonstrate through an example, and the idea here is that suppose at this node, sixty units are available. So, I start with the first arc the twenty units is the capacity. I will just flow twenty units along 1 2, twenty units here and twenty units here, because the remainder is 20 left here now of this one 60; so, I will have 20. So, this is my first set of allocations. Then I go to node 2 and node 2 you see had demand of minus demand of 15.

(Refer Slide Time: 17:07)



So, 5 units will be surplus. So, the idea is that I will observe the fifteen and then plus 5 will go here. Now, look at twenty here which is coming to node 3. Node 3 has no demand. Therefore, I will send this twenty to the first arc outgoing from here twenty, and so, this becomes 25 write this gets used up.

Now, you come to node 4 and node 4 has ten units available; so, that means the total amount available here becomes this things. So, I should have used it. So, yeah, right, so, I should look the number here. So, 10 plus 20 20, 10 I send here because this is the maximum that you can send along arc four 6, and so, 30 will go here, sorry, 20; 20 came from here; 10 were available here.

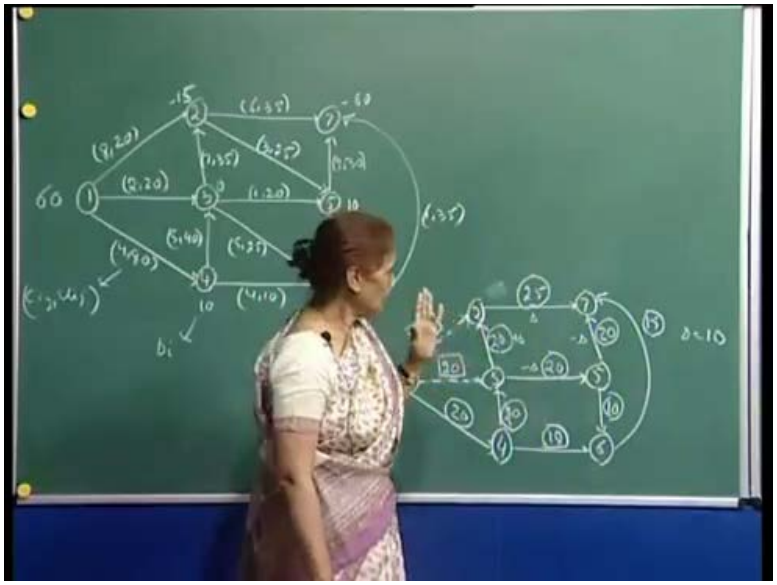
So, in all the 30 gets distributed, and then, I come up at node 3. Again this one is observe. So, here this is 3 5; 3 5 the capacity is 20. I send all the flow all the units available here along the r 3 5, and then, it 3 5 you have ten more units. So, we will send twenty here, because why twenty here? I could have send, because this was minus 5, yeah, then this up to you. So, write 1 because capacity here was 30. I could have send all the 20 here and the 10 here. I could have send 30 here and because the demand here is 60.

So, I could have done that 30, and then at node 6, 5 units got absorb, but anyway, I chose to read this to redistribute, so, 20 10, and there is another reason, because I want to show you how to reduce this feasible solution to a basic feasible solution. Anyway, so, does not matter. So, here, then you have 5 units available required. So, from 10 5 are left, and so, along this arc, you will send how many? So, let us see what is happening. Yeah, 25 is coming from here; 20 is coming from here; 14, 45, then 10 came here and 5 over left over because 5 got demanded.

So, 15, so, 15 go here, 45 plus 15 – 60. So, this is a feasible solution. All is so ad-hoc way. You can, you can sit down and program your own way of doing it how to compute the ad-hoc method. By ad-hoc method, you get a feasible solution, and now, because we want to get rid of the cycles, remember, a spanning tree solution will have no cycles present in a. So, we will one by one try to redistribute the flow along the cycles. So, if you look at the first cycle here 1 3 2 1, so, I cannot do anything here, because the capacity is 20 and the capacity here is also 20; here it is 35, but for the increasing, the flow here along the cycle I cannot, I will have to increase the flow here also, I cannot do

it. Therefore, for this cycle what I'll do is since both these arcs are at there upper bounds, I will, I will, I will, rename these arcs 1 3, 1 2, 1 3 as non-basic arcs, non-basic arcs at their upper bounds. Remember the idea is that we are, I just want to have, how many nodes are there? Seven. So, I should have 6, I should have 6 arcs which are in my basic, which form my basic tree.

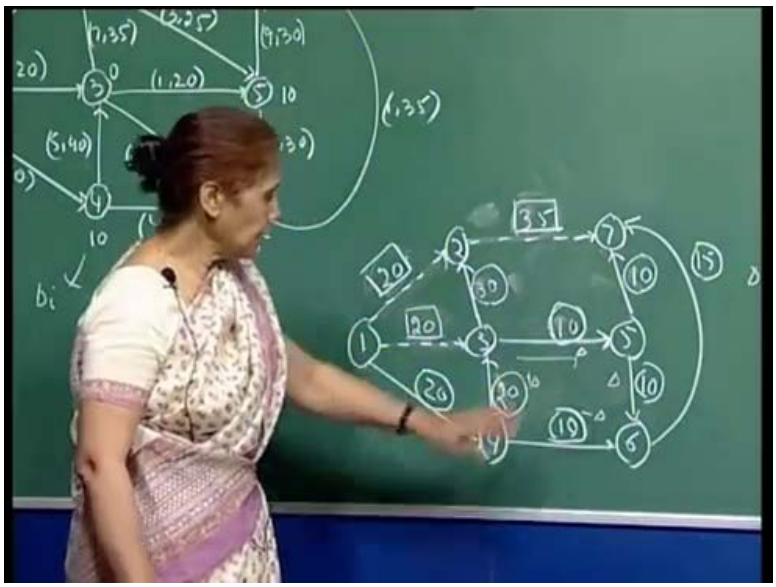
(Refer Slide Time: 20:39)



So, I will rename, I will just put instead of this, I will put a square. To indicate that this is a and I will write a broken this thing. This a broken arc to indicate that it is not a part of the my basic tree. So, here also this is 20 and this will be broken; so, that is it. Now, we want to go look at, therefore, the, the, both these cycles are not there anymore. Now, you look at the cycle or you look at the cycle 2 3 5 7, 2 3 5 7. So, here, arc 3 2 has capacity more than what is. Therefore, if I try to flow delta units here, then this will be plus delta I can do it, then this will be minus delta and this will be minus delta. This way I can redistribute the flow. So, what is the limit for delta? It comes out to be twenty, and since the idea is to get rid of, as many non-basic, as many arcs is possible in the sense that I should only have seven of them left. So, I will choose delta to be twenty, and so, both these arcs will go away, and so, the flow here will become four, but I cannot choose delta to be 20 because this is 35, so, only 15, and here, this is 10.

So, I can choose delta to be 10 only; so, that means, yes, and once I choose delta to be 10, then this arc will reach its upper bound and I can then call it a non-basic arc, because I am searching for, 6, 6 basic arcs which are, which can form my basic feasible solution.

(Refer Slide Time: 22:33)



So, delta is 10. Therefore, this flow becomes 30, and this now becomes a non-basic arc. So, this is 35, I will do it this way and this is a broken arc. So, here, the flow becomes 10, and here also the flow becomes 10 units. So, this is what. Keeping upper bound constraints and maintaining flow conservation, that law anyway be maintain. If I reallocate the flow along cycles, so then, I have this. So, I have manage to get rid of 3 arcs or at least rename 3 arcs as non-basic.

Now, you look at this cycle here. So, at this cycle, see what is happening is, yes, I can, I can, try to increase the flow here. The upper bound here is 40 that is no problem. Then I can reduce the flow here by 10, and what about 5 6? 5 6 also, if I increase the flow along this, then it will go plus delta, plus delta, minus delta and plus delta. So, this arc will get remove because delta is 10. See 5 6 the limit is 30. So, fine, for 3 5, it is 20; so, delta can be 10, and 4 3 just plenty of scope because its upper bound is forty.

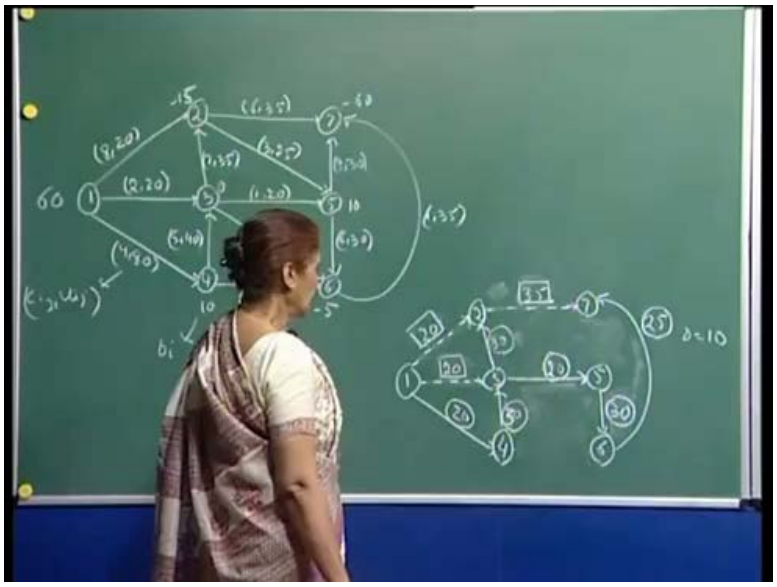
So, I can choose delta to be10 and again you have the choice of either making this non-basic or this non-basic, because this will be non-basic at its upper bound and this will non-basic at 0 level. So, let us do that.

(Refer Slide Time: 24:27)

So, we will remove this. This arc goes out, and so, this will become 30, 30, and this will become 20, yeah, sorry, no upper bound, this is remain satisfy. So, this is 20, and here, the flow is also 20 but I am leaving it as a basic arc or I can, I can, decide to, so, let see how many do we have - 1 2 3 and I need the 3 more. Therefore, I will have to keep this later, because this cycle I have to remove one arc, I have to get rid of that cycle.

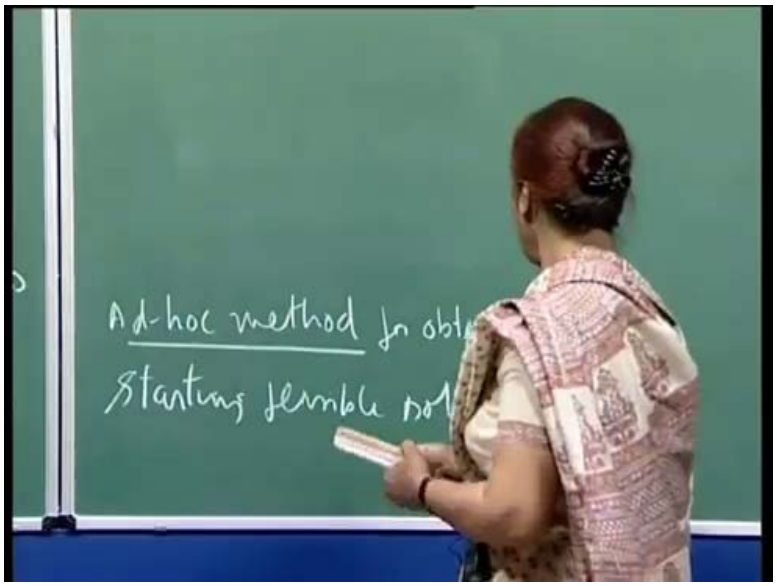
So, therefore, I could not have made this a non-basic arc also at its upper bound. So, this is the new thing, and now, we look to get this cycle. So, 6 to 7 the upper bound is 35 and rewrite. So, I can, I can, increase the flow along this direction by delta. Then this will be minus delta and this will be plus delta, and again, delta can go up by 10 because the upper limit is 30.

(Refer Slide Time: 25:47)



So, this arc will go out, it will remain it will become non-basic and this flow will be 25 and this flow here will be 30. So, you see if you remove these arcs, then you have 1 2 3 4 5 6, six arcs they form your spanning tree and this gives your starting feasible solution.

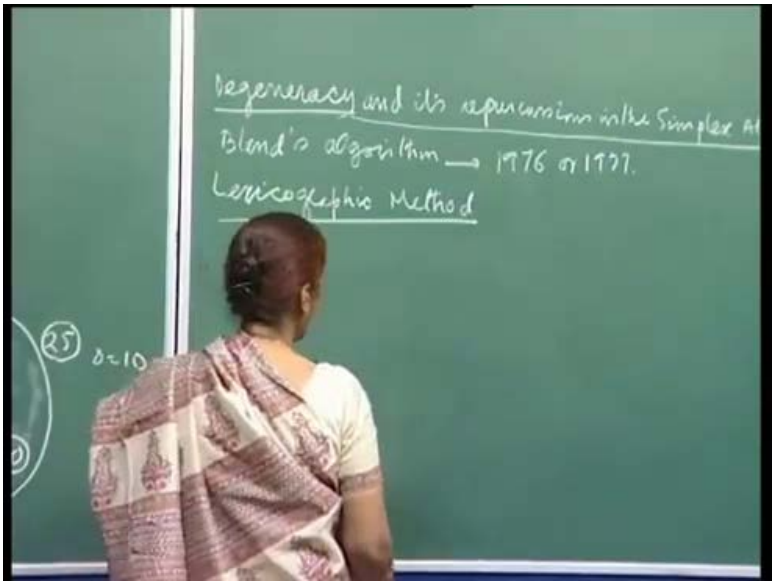
(Refer Slide Time: 26:27)



So, this is an Ad-hoc method and one can do but mostly the software that has developed for min-cost flow problem, they, they, program phase I because easier to program it. Here, you can see that we will have to use lot of judgment at every point, how to do it though one can probably program this also, but anyway, I just wanted to show you that

you can get a starting feasible solution, yeah, so, that takes care of your starting feasible solution. The next step is now what you want to talk about is degeneracy. Remember that are the second issue that we have to tackle here. Let me here begin from the very beginning.

(Refer Slide Time: 27:10)

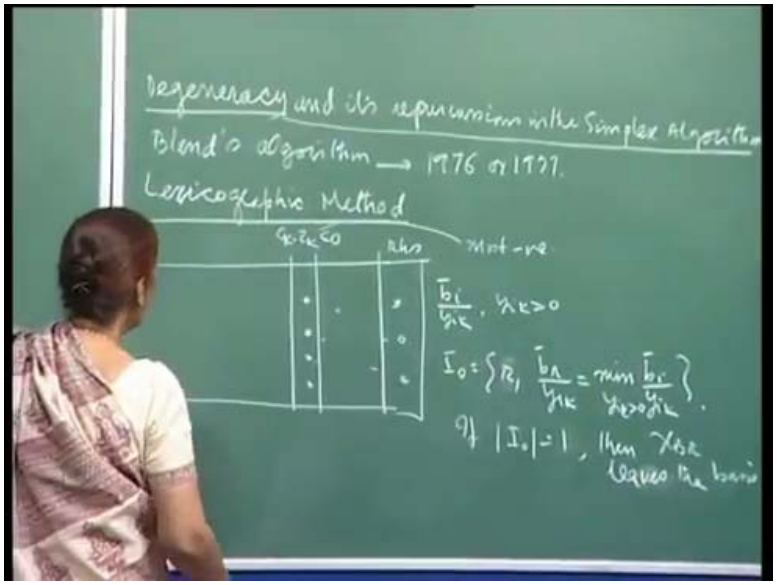


So, first, I will talk about degeneracy in the simplex algorithm degeneracy and its repercussions, repercussions, in the simplex algorithm. So, I did talk to you about say what I gave you first the bland's method and this I think came around I must have given you the date exactly in my talk. When I was talking about the bland's algorithm 1976 or 77 or, or, I am not very sure, but anyway, one can check that. So, this was given by bland but I should have any discussion on degeneracy would be incomplete without the lexicographic, because that concept still is used maybe different ramifications of that concept is still there. So, I thought I should talk about lexicographic method.

So, what does lexicographic? What does it say lexicographic method? So, the idea here is that you want to make the choice of the outgoing variable unique and of course, one can say may be you toss a coin maybe you can, but it has to be where you show that the particular choice will lead you to, will lead you to, finite number of steps of resolving degeneracy; that means you should not cycle.

So, this is what we mean by saying that the choice of the exiting and the incoming variable should be unique in such a way that you can say that the number of degenerate pivots will be finite; I mean a sequential or successive degenerate pivot should be finite. This is the idea. So, lexicographic was much before bland came up with this. This was write in the early sixty's, it was proposed.

(Refer Slide Time: 29:48)



And the idea here is, see, I will draw the diagram and then write down the rules. See what is happening is you have this right hand side vector, and suppose you choose this variable this variable to enter the basis. So, your c k, c k, minus z k is less than 0 and we are saying that this is a most negative. So, our incoming choice just has dantzig algorithm said - we choose the most negative one to enter the basis. Then, what was happening was that we looked at the positive entries here. So, this is the kth column.

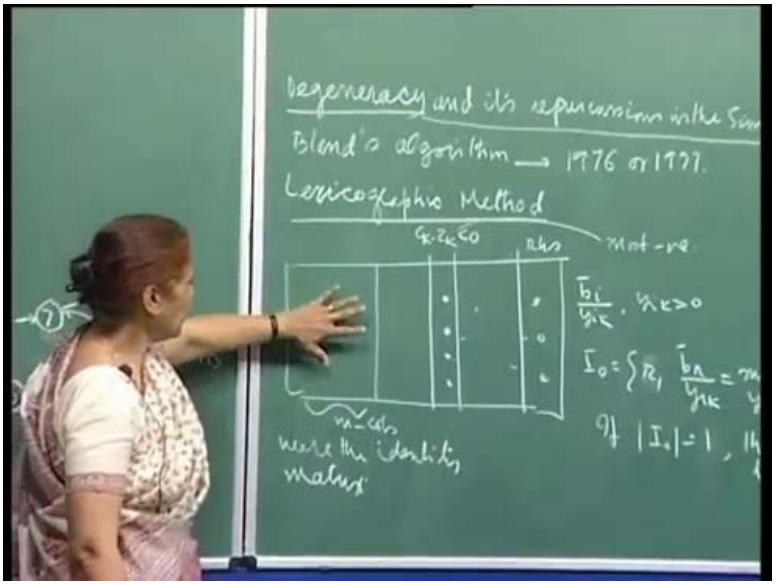
So, we looked at the positive y i j's and then took the ratios. Therefore, we first step was you took b I, b I, b I, y the, this upon y i k. So, we took the ratios for where y I k is positive, and then, we said that or let me write the index set here. So, we said that let I naught be equal to all I or all r consists of all r such that b r bar upon y r k is minimum of b I bar upon y I k, y I k, positive.

So, I not was this. Now, if r was unique, if I naught has only had only one index in it which is r, then, so, if I naught is equal to r, then, sorry, here, it will mean 1, then x b r

leaves the basis, l e a v e, leave the basis, but, the problem came when you had more than one index in the set I naught, that means there was a thigh of course, for this, for this and this there was a tie all 3 ratios were the same.

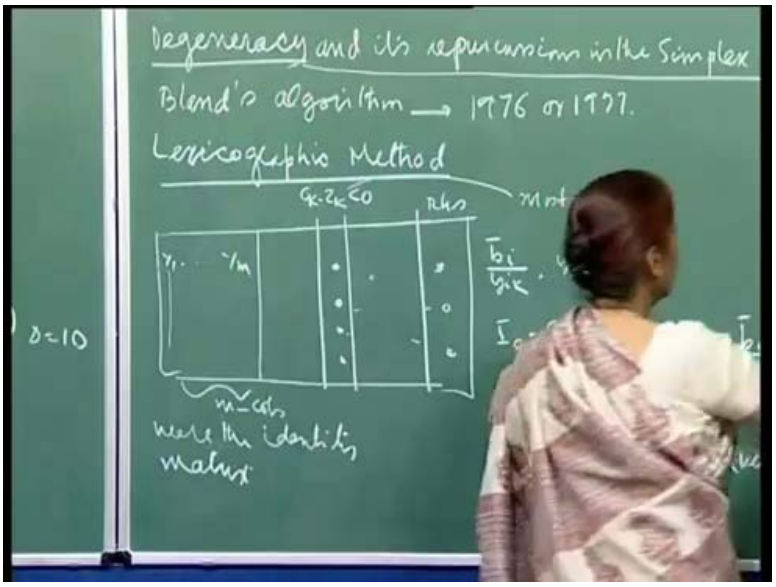
So then, how do you decide which one exits the basis? This is where and because once you do the pivoting, then in anyone and the remaining 2 variables will become 0, then your values will be 0, and so, you can cycle. So, one, one, one, wants to avoid that. Therefore, bland gave you one rule which was that if you have more than 1 and choose that smallest index; that means out of the 3, this is the smallest basic variable in terms of its index. Then, you, it allow this 2, leave the basis, so, bland gave you this one.

(Refer Slide Time: 32:53)



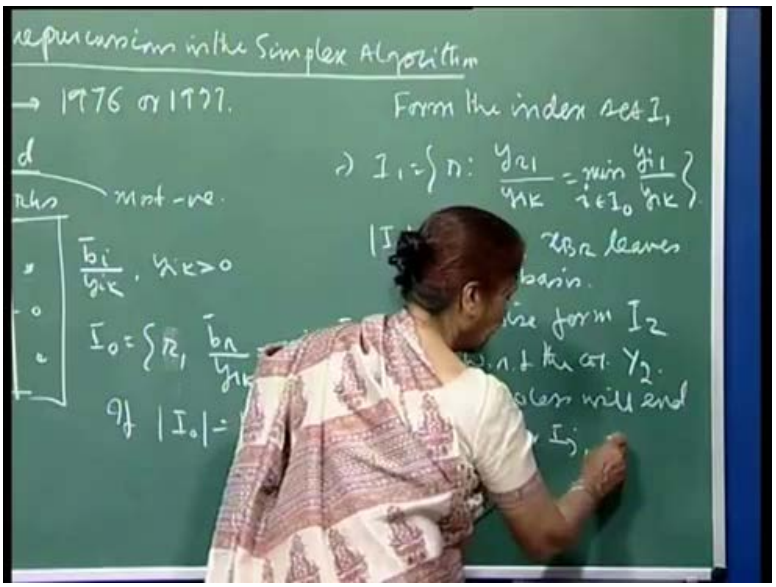
Now, before that the prevalent rule was, see suppose these for m columns, these m columns for the identity matrix in the beginning. So therefore, this initially this was I that when we started the algorithm, suppose your starting basic feasible solution was an identity matrix. At this is happen if you have slack variables consisting of a making of your basis, then this was the identity matrix.

(Refer Slide Time: 33:25)



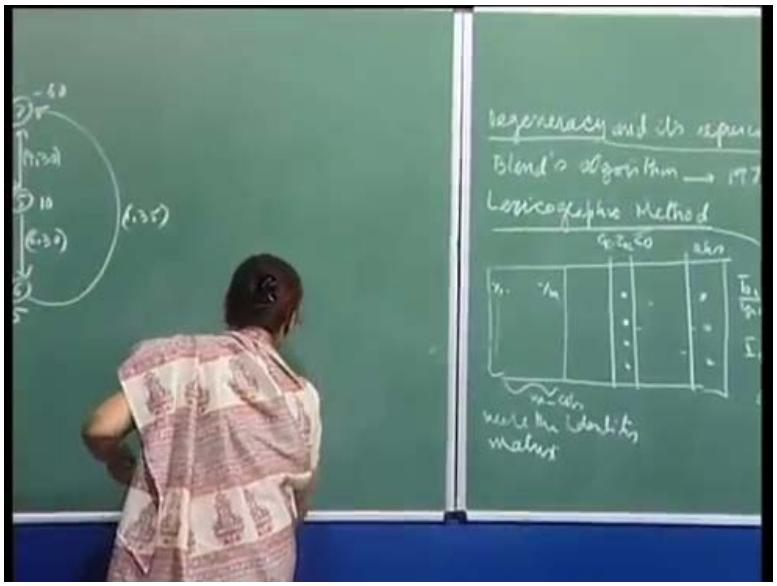
And now of course, so, you have y 1 to y n. So, it some intermediate iteration, these are the new columns they have been. So then, they said that suppose, so, now, we have I naught, you have more than one index here, then you take the corresponding element here and divide by this. So then, you take the ratios for this column from here; that means, sorry, first you chose, you took the ratios for the y i k is positive you took the numbers here and took the corresponding ratios they were thighs, then you take the now the ratio.

(Refer Slide Time: 34:05)



That means you should form form the index set, index set, I 1 such that I 1 consists of r, where now you have y r 1 y r 1; that means the rth is this thing from here divided by y r k. This is minimum of y, y i, 1 upon y r k; I belonging to I naught. The definition is that for all the indices that are present here. You take those components corresponding components in the first column here and then take the ratios from the corresponding y i k's, and then, again find out the ratios and then choose the minimum one. Now, seen rule here again if this is 1, then x b r leaves the basis; otherwise, you then, otherwise, form i 2 with respect to the column, respect to the column, y 2. So, do the same thing; that means whichever indices are there in I 1, for which, the ratios is the same there is a thigh. For those ratios, you will choose the y 2 column, take the corresponding ratios, and then, again, find the minimum one. So, this process goes out, and why will it end, because you see if the things continue to thigh, then the process will end, end, for j less than or equal to m. Process will end for, I should say for n for i j - where j is less than or equal to m.

(Refer Slide Time: 36:22)



What I am trying to say here is that taking the ratios for column wise, the process must end much before we reach the mth column. Suppose, it does not happen, then what would it mean that the index set I m has cardinality more than one; that means you I do not have a, I have not come to unique choice of deciding which variable to leave the basis. So, this means at any 2 rows with indices I m are multiple of each other that will imply, because you see what is happening is that if index set of I m is more than one, say suppose r 1 and r 2 are 2 indices in I m, then what does it mean? That the ratios are the

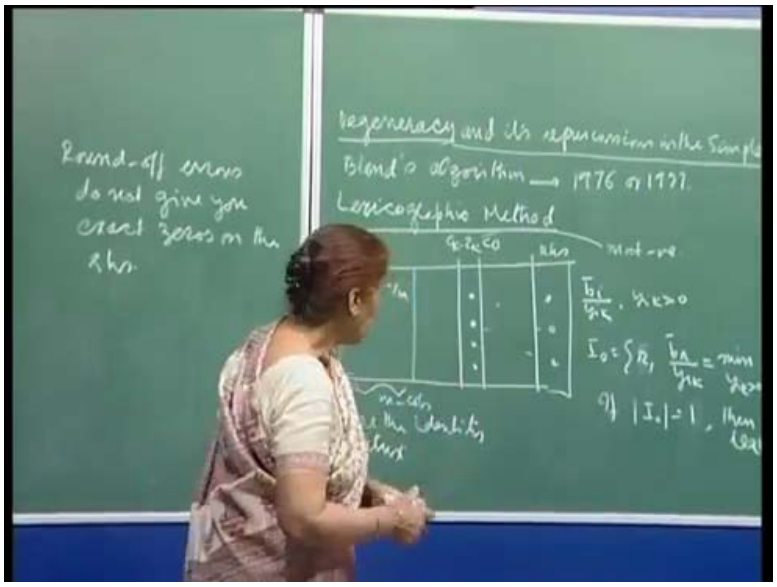
same for all because I have reached up to here, and so, for each of the columns, the ratios have been the same. So, y r 1 j upon y r one k is y r 2 j upon y r 2 k corresponding to this column, I am taking the ratios.

So, then j varying from one to m which would imply that y r 1 j is y r 1 k upon y r 2 k y r 2 j, because these numbers are non-zeros remember positive. Therefore, you see and this is 2 for j varying from 1 to m, which means that the r one th row is a multiple of is a multiple of the r 2 th row and these two rows are the rows of b remember, because we said that these columns will present the, initial, our starting basis was here.

So, these rows are but, that means here, if r 1 th row is a multiple of r 2 th row, the 2 rows cannot be linearly independent but that is not correct, because rows of be are linearly independent. Therefore, this process of taking the minimum ratio must end much before we reach the mth column or at most at the mth column, that is, when I have a unique choice of deciding, which variable should leave the basis, and that is the whole idea, because we said that we want to break the cycling process by making a unique choice of incoming variable and the exiting variable, but as we said that these are all very time consuming things.

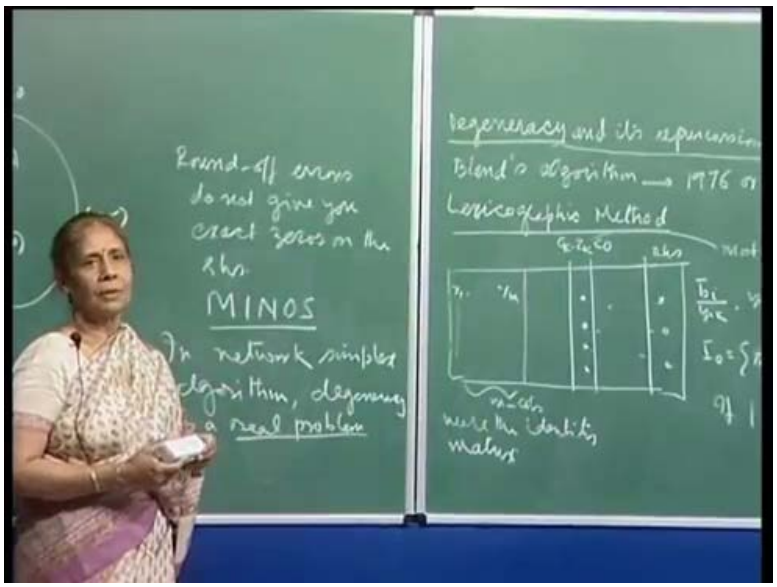
So, now, you see, this is fine; this use to work fine, but again, you can yourself see that to program this would, and if you have to do this exercise at every iteration, it will be, it will be very expensive. So, that is why bland came up with his idea which is was very easy to implement. Bland's idea did not require any such calculations, you could very simply implemented. It could be a program in any software for the simplex algorithm, but then, work done out is it was notice when the bland's algorithm got used all that the number of iterations became very large, because the long paths of degenerate pivots. You could continue to have though of course, you prevented cycling of course, which is better than having a long set of, long path of these degenerate pivots, but then, same thing, I mean here, you had to do many more iterations. Here, it was expensive to, to, you know, work out every iteration of the simplex algorithm. Then it turned out that what was said is that because here you do for when you substitute your right hand side, compute the basic feasible solutions, we divide by the determinant of the basis, and so, because of the rounding of errors, so, I should write it out somewhere here. So, the round-off errors play a role.

(Refer Slide Time: 40:19)



Round-off errors do not give you exact zero's on the right hand side. You will seldom c zeros here exactly, because of the, because you know that the computers work with finite algorithm, and so, what was spelt was that this, this, will really not happen when you are dealing with the general linear programming problem and solving the by the simplex algorithm, because, this right hand side will appropriately get (( )), because you will not get zeros, and so, you will not have a degenerate solution, and therefore, there will be no cycling. This is the empirical feeling and this is fine .

(Refer Slide Time: 41:16)



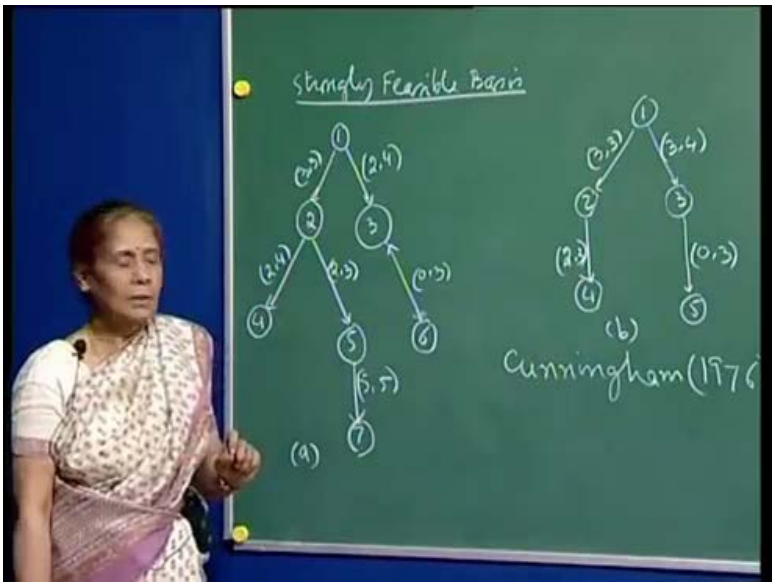
But, there is also I think this is a software developed by, I think some, some, Canadian people so which, which, sort of better the hand side appropriately so that you do not get degeneracy.

See this is of course, a very theoretical in comprehensive method, but as I said it will be very expensive so this program takes care of degeneracy and therefore, cycling in the simplex algorithm. Now, for the network simplex algorithm, store is different, because for the network simplex algorithm, you do not the determinant of the basis is always one or minus 1. Therefore, you do not the round of errors are not, do not occur there and your, your, basic feasible solution will always the plus minus of the b i's, and therefore, the zero's will occur.

So, in the, in the network simplex algorithm, degeneracy is a real problem. So, we will say in, yeah, this is the last statement I make this is a network simplex algorithm, and a network simplex algorithm, degeneracy is a real problem, real problem; I mean that it will occur because I have not yet talk about the assignment problem or the shortest path problem. We will, I will show you that you see, even in the shortest path problem that we discuss, it is not necessary that a shortest path will have n minus 1 (( )). It may have a few other n minus 1. So, you are going to have a basic feasible solution which is degenerate. Similarly, I will some time mention the assignment problem which is highly degenerate. Therefore, in network simplex algorithm the, algorithm, degeneracy is the real problem, and so, we have to calculate.

And so, for the transportation problem for example, I have mentioned it that you should tried to I gave you one excellent perturbation method for handling degeneracy that I said you should applied Bland's algorithm also, and you will see that you may end up with more iterations, then the excellent perturbation method. So, we have to then compact degeneracy in the network place algorithm, and will, because of the structure again, we will able to give you perturb methods, then bland or lexicographic method. So, we discuss that after this.

(Refer Slide Time: 43:54)



So, Cunningham in 1976, maybe I should give the name here. Cunningham in 1976 came up with the idea of working with strongly feasible basis. So, let me explain what we mean by that a strongly feasible basis definition, this is a rooted, rooted, spanning 3 such that all arcs with 0 flow are pointing. This is pointing towards the root pointing towards the root and all arcs, I mean in the tree expanses, the spanning is the span rooted spanning tree so that all arcs with 0 flow are pointing towards the root and all arcs at their upper bound are pointing, pointing, against the, pointing in the opposite direction, in the opposite direction.

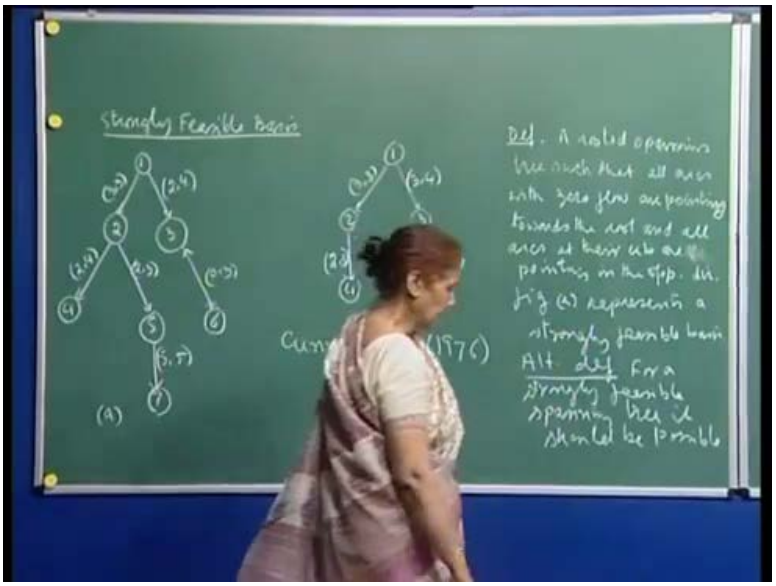
And you can see that. So, I will give an example what we mean by this. See this is, this is an example. So, you let, if I call this a and this I call b, then represents strongly feasible basis. You see because here this is an arc which is saturated and it is pointing away from the root. This of course does not come into the definition because it is not saturated and the flow is positive. Similarly, here, again this was saturated, so, the arc is pointing away from the root.

Then here, this is this is not saturated. Here, the flow is, so, this is not, I have to say that this should be this way. So here, this is 0 flow, then it has to be pointing towards the root, and the idea is that another way to define strongly feasible basis this that from every node of the tree, you should be able to send positive flow to the root and you see that immediately from here for example, I can send a positive flow of 2 units because you

already have 2 units here. So, this will flow will get reduce; it is possible to do it. Then from here, because again you have 3 units in this direction, I can send 2 units here. Similarly, from here, because that is saturated, I will reduce the flow and I can send 2 units here again along this arc.

Similarly, from this point onwards because the flow is 0, upper bound is 3, I can send are at most 3 units and then but, here and I can send 2 units. Therefore, you come to 6 to 1, you can send 2 units.

(Refer Slide Time: 47:14)



So, a represent, so, figure a represents a strongly feasible basis, yeah, I have not given you the, see idea is that it is the feasible solution in the sense that with the demands in the supplies are net, and after that, this is the additional condition we are imposing, but, figure b does not represent strongly feasible solutions spanning tree, because here, you see this arc has 0 flow and it is pointing towards away from 1. See, therefore, from 5, I cannot send any flow 2 1.

From node 5, I cannot send any positive flow. So, an alternate definition is, definition is that for a strongly, strongly, feasible spanning tree, it should be possible, it should be possible to send a positive, positive, flow from any node to the root node. So, this is an arc net and which one can very easily check. One can be write a small program to always check whether this is this right. Now, then the second thing is how do you maintain at

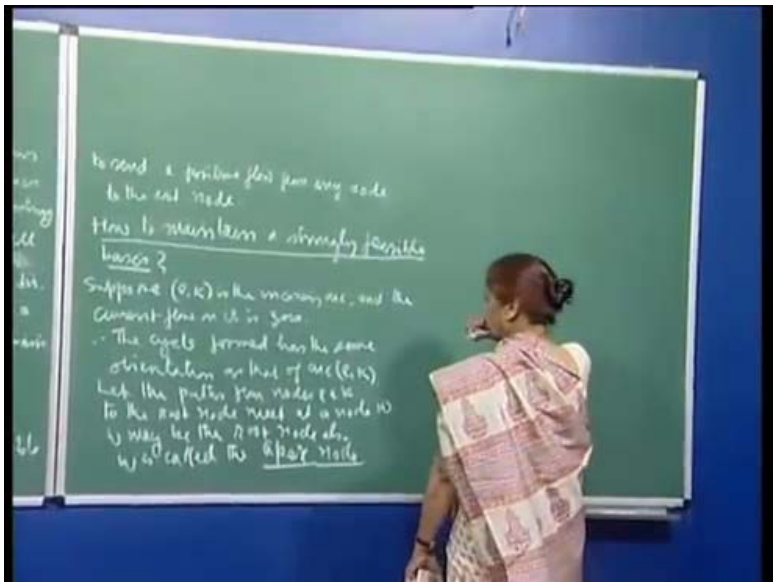
every time, because if you have to bring in a basis, arc into the basis and remove an arc then, so, how to maintain, that is the next step.

(Refer Slide Time: 49:23)

How to maintain a strongly feasible basis? How to maintain a strongly feasible basis? So, the idea here is that suppose I will first explain in words suppose this is an arc which is coming in into the basis. So, suppose 7 6 is an arc which has to enter the basis, then and suppose it is at 0 flow, at 0 unit the flow on the arc 7 6 is 0 currently. Therefore, you are going to, this will be the orientation of the cycle and you will increase the flow by delta here. Now, this increases delta. Then here, it will be plus delta; this will be minus delta; this will be plus delta, because it has the same orientation; this will be plus delta and this will be plus delta..

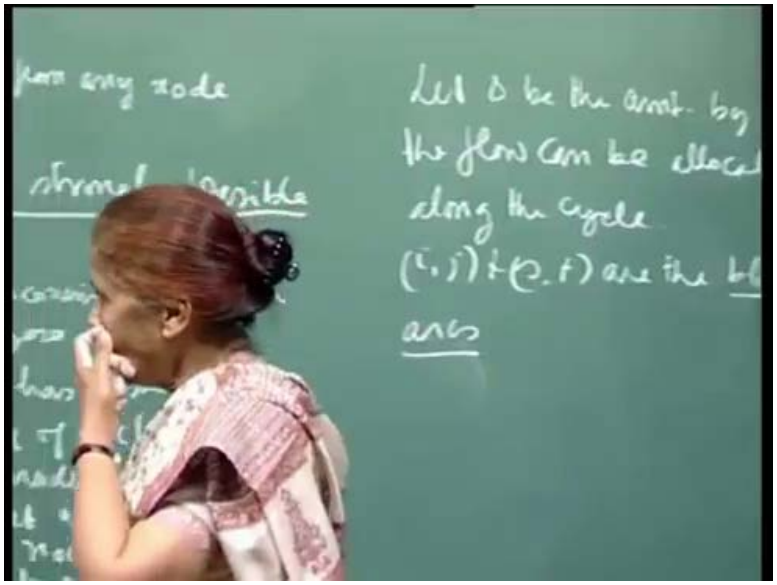
So if this is an incoming an arc, then this is what you had. Now, just see, so, first of all we define what is called an apex node. So, I will write it down the definition. The idea is that if this is an arc which is coming into the basis, then from both these nodes, the tail and the head of the incoming arc you have path up to the, you have a path up to the root node. So, if this paths meet somewhere before the root node and that will be the apex node. Otherwise, root one is the apex node. So, we will first let me make a few definitions here.

(Refer Slide Time: 51:13)



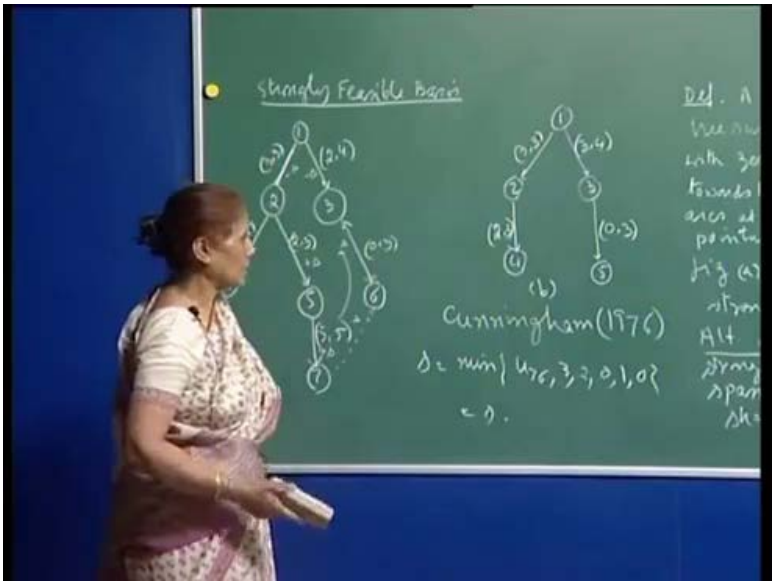
So, what we say is that suppose, so, I will give you the rule for, suppose l k is the incoming arc, arc, and the current flow on it is 0. Therefore, the cycle formed, formed, has the same orientation as that of the arc l k, as the same orientation. Then, let the paths from nodes l and k to the root node meet at, at, a node w; w may be the root node also when w is called be apex node, w is called, w is called the apex node, say apex node, and then, to this rule was simple.

(Refer Slide Time: 53:13)



So then, determine and suppose, now, let, let, delta be the amount by which the flow can be, can be, allocated along the cycle or actually re allocated along the cycle. So, and you can say and suppose that i j and s t are the blocking are maybe, are the blocking arcs may be more than 2 blocking arcs.

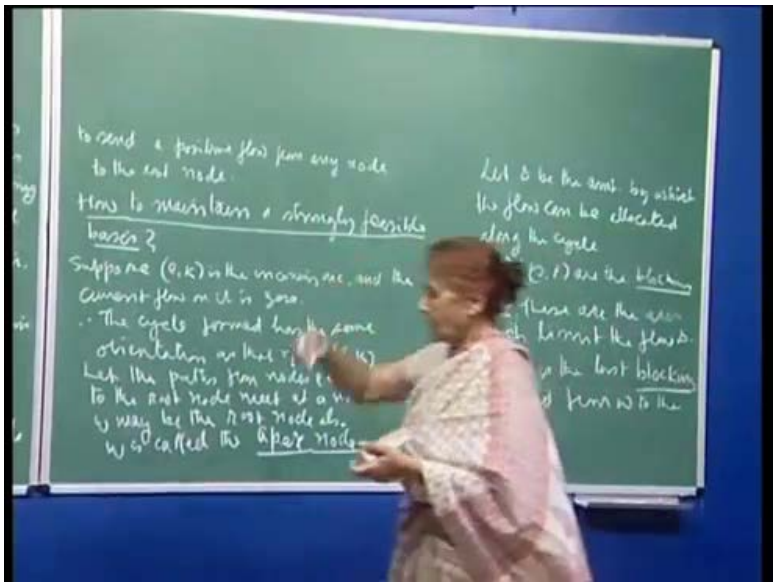
(Refer Slide Time: 54:17)



Blocking arcs we mean the arcs which will, yes, the arcs which will limit the value of delta. So, for example, here, the value of delta will be determine by, see, delta would be what? Minimum of u 7 6, because the flow on this cannot go beyond the upper bound limit, and here, for example, the flow can be go up by 3 units only. Then here, the flow can decrease by 2 units only. So, I am just writing out like this, and here, the flow cannot increase at all because is already at its upper bound; so, it will be 0.

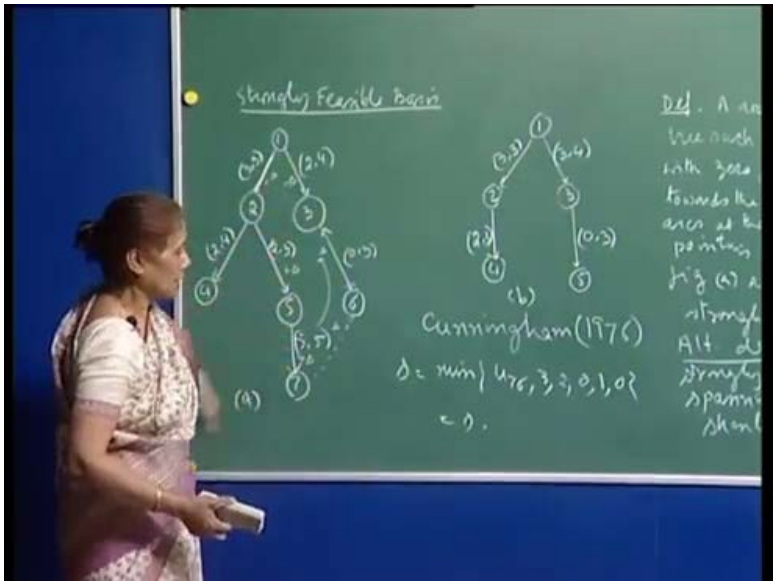
Then here, it can go up by 1, and here, it can go up by, it can again go up by 0 only. So, in this case, it is 0 and the blocking arcs are this. So, the blocking arcs, therefore, in this case, the blocking arcs are 1 2 and 5 7, because they limit the value of delta. The delta cannot be anything positive here. So, these are the blocking arcs.

(Refer Slide Time: 55:22)



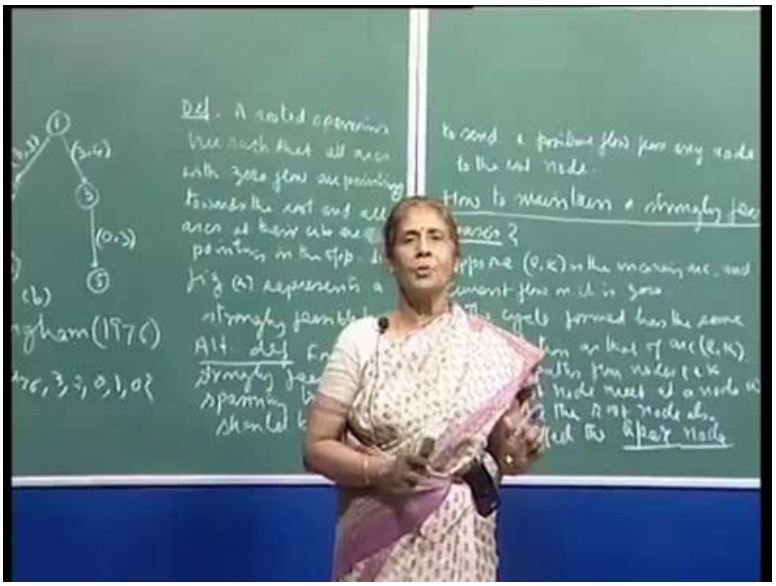
So, the idea is therefore, now I am define for you what are the blocking arcs. These are the arcs which limit flow of delta. Now, what it says is that choose, so the exiting arc this is the rule. Finally, they are giving exiting arc. So, it has to be unique is, so, this is, is, the last blocking arc, last blocking arc, when the, when the cycle, c y c l e, the last blocking arc, last blocking arc reached from w that is the apex node, reach from w to, to, the node. We had, we said that the arc that is incoming is l k to the node l.

(Refer Slide Time: 57:09)



That may explain that means it is the last blocking arc. See here, what we are saying is that this will be, if your coming has the from w, this is your apex node now. So, when you're following the path from the apex node to the the incoming arc, then the last blocking arc; that means by this definition, this will be the arc. So, the flow will not change; that means this will become a basic arc. If you want, so, make it at 0 level, you want to make it basic, then this will be the exiting arc.

(Refer Slide Time: 57:40)



So, here, this is the last blocking arc. So, one can make it more here this is say that exiting arc is the last blocking arc reach from w to the node l, reach from w along the path. We can say that here I can make it more w along the path, along the path to node l. So, it is a last one if you do it, then what you have it. So, the remaining, so, the new tree, new tree, is again strongly feasible strongly feasible, yes. So, I will show you some more this thing, and of course, the question would be how do you then get a starting strongly feasible solution. The phase one tree that you got is a strongly feasible solution, you can check for yourself, because these, the artificial variables, artificial arcs do not have any upper bound limits on them.