Advanced Operations Research

Prof. G. Srinivasan

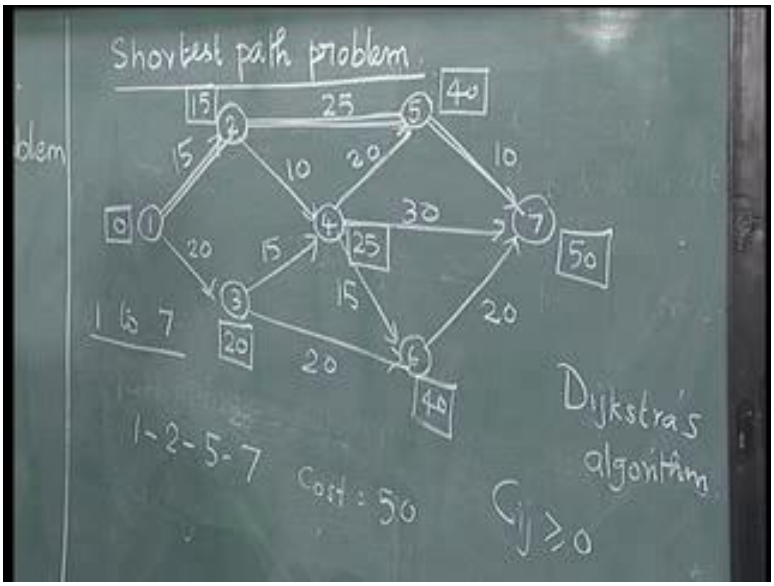
Department of Management Studies

Indian Institute of Technology, Madras

Lecture - 20
Shortest Path Problem

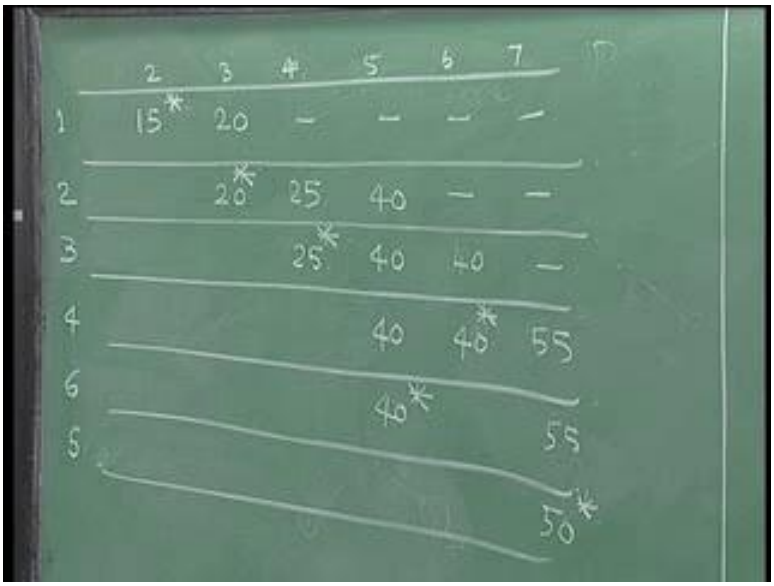
We continue our discussion on the shortest path problem. In the previous lecture, I explained the shortest path problem using this numerical example.

(Refer Slide Time: 00:22)



The problem was to find out the shortest distance and the shortest path from 1 to 7. We have already said that this arc could represent a cost or a distance or a time. We would like to find out the shortest path and the corresponding sum of weights associated with the shortest path. We started with a 0 here, and then we first found out the minimum distance that is reachable from this. 2, is reachable at 15, 3 was reachable at 20 so we picked the minimum and labeled this as 15. From 2 we proceeded to subsequently label 3 at 20, 4 at 25, 5 and 6 at 40 and 7 at 50. We found out that the shortest path was 1 to 2, 2 to 5, 5 to 7 with total cost or distance equal to 50.

(Refer Slide Time: 01:21)



We then proceeded to explain a tabular form or a labeling algorithm to do the same thing that we did on the network. This rudimentary algorithm is the Dijkstra's algorithm and we now present a labeling version of the Dijkstra's algorithm. Here 1 was the source node so we start with 1 we write the rest of the nodes here from 2 to 7. We go back to this network to see which ones are directly reachable from 1. They happen to be 1 to 2 with 15 and 1 to 3 with 20. We write 15 and 20 here, 1 4, 1 5, 1 6, and 1 7 do not exist. Therefore, we write a dash here or we can write infinity, saying that at present the distance is infinity. Among these values, there are only two of them, between these two we pick the one which has the smallest numbers which is 15 and mark it with the star or with some form of a label.

Since node 2 is marked we write 2 here and then we proceed to this network to find out what are all the other nodes that are reachable from 2. They happen to be 4 and 5. Now 2 to 4 with 15 plus 10 is equal to 25 and 2 to 5 with 15 plus 25 is equal to 40. We look at these values. If you look at 4, 4 is reachable with 15 plus 10 is equal to 25. Earlier, the distance was infinity, 25 being smaller than infinity, we changed this value to 25. Similarly, 40 being smaller than infinity, the value are updated here from infinity to 40. Now, 20, that was a distance from 1 to 3, remains as 20 because 3 is not reachable from 2, so 20 remains as 20. We have three values 20, 25 and 40. Out of these three values, we pick the minimum one, which happens to be for 20 and then we mark a label here.

Since 3 is marked, we write 3 here and then go back to this network to see what are all the things that are reachable from 3. They turn out to be 3 to 4 and 3 to 6. 3 to 4 is 20 plus 15, is 35. 35, happens to be more than this 25, so we retain this 25. Now, 5, is not reachable from 3, so 40 remains as 40. 6 is reachable from 3, 20 plus 20 is equal to 40, 40 being less than infinity, infinity value is changed to 40. Again, there are three values; choose the minimum which happens to be at 25, which also happens to be for node number 4. So, node number 4 comes here and this process is repeated till node number 7, which is the destination node; it is labeled and the value is equal to 50.

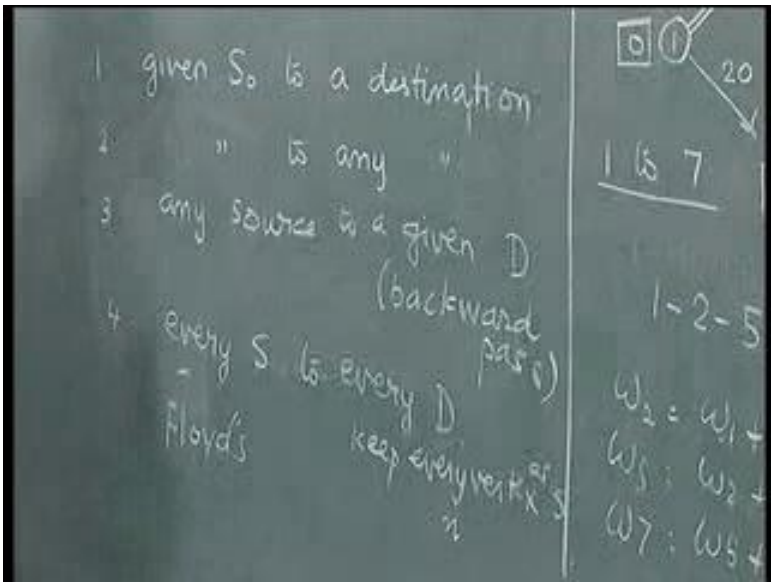
What we have done here is we have systematically written this. This labeling version of Dijkstra's algorithm is far easier to understand and implement, if one was to write a computer program to find out the shortest path on the network. The same 50 which happens to be the shortest distance, obviously, the same 50 we have got here. While for this, we said it is 1 to 2, 2 to 5, and 5 to 7. The motivation came from the fact that 0 plus 15 is 15, 15 plus 25 is 40, 40 plus 10 is equal to 50. We have a path, such that every label for every i j in that path, the label corresponding j is the sum of the label corresponding to i, plus this C_{ij}.

For example, you could, if we call this as some w_2, which is the label corresponding to 2, then w_2 is w_1 plus c_{12}, w_5 is w_2 plus c_{25} and w_7 is w_5 plus c_{27}. There is a path from 1 to 7 and we got this 50. How can we identify this path from this table? This is something we need to do. What happens is the final value is 50. There is a change, move upwards, so this change is happened from 55 to 50, because we passed through node number 5. The first thing, we have to do is write 7, which is the destination. Since, there is a change here from 55 to 50, that change has happened because we have passed through this 5, so, we write 5 here.

Now, come back to this, we will be at 5, now, move upwards again. Here, there is a change from infinity to 40 and this change has happened, because we had moved from 2, so we write 2 here. Then this is the change, and then come back here to 2, we moved here, there is a change and this change happened, because we labeled this 2, so we write 2 here and then we write 1 here. To repeat, we got this 50, there is a change from 55 to 50. We represent the first destination node here. Then there is a change from 55 to 50. This change has happened because we moved from 5. So, 5 come here, so automatically, we go to node number 5, which is 40. Move upwards, there is

a change between this infinity and 40, because we moved from here to the labeled node 2, so we write 2 here. Then, we come back to this 2, we realise there is no change; therefore, it is 1 2. The path becomes 1 to 2, 2 to 5, and 5 to 7, with weight equal to 50. This is how we explain the labeling algorithm for the Dijkstra's and the way to get both the shortest distance as well as the shortest path.

(Refer Slide Time: 08:21)



What else can this Dijkstra's algorithm do?

When we look at shortest path so far we have seen shortest path from a given source to a given destination. The second type of problem that we can think of is from a given source to any destination. Let us see if the Dijkstra's algorithm can solve this problem as well. If we assume that our given source is 1 we can easily show that these label is that we have written here, not only helps us in getting the shortest distance from 1 2 7, we have actually solved the shortest path from 1 to 2, 1 to 3, 1 to 4, 1 to 5, 1 to 6, 1 to 7. We can show that the shortest distance from 1 to 2 is 15, 1 to 3 is 20, which is obvious for 4. It is 25 for 5 and 6 it is 40 and for 7, it is 50.

The Dijkstra's algorithm not only solves the first problem whereas it helps us to find the shortest path and the shortest distance from a given source to a given destination. It also solves the second problem where it does it for a given source to all destinations. With one pass of the algorithm, we have actually solved both these problems. There is a third problem which is like

from any source to a given destination can be seen as a third problem. This can be done in more than one way, one of which is to do a backward pass of the algorithm from here to here.

Remember, what we did first is a forward pass from 1 to 7. We can actually do a backward pass from 7 to back, if we do that, we will be able to find out the shortest distance from every one of this to 7, because a backward pass implies that we are actually trying to find out the distance from 7 to every other node in the network. Backward pass of the Dijkstra's algorithm would help us to find the shortest path from every node to a destination or from destination to every node on the network.

The fourth problem is between every source to every destination, which means, we wish to find out something from 2 to 6 or 3 to 5 or 4 to 7 and so on. One of the ways of using this algorithm is to do this, keep every vertex as the source and run it n times if there are n vertices. Running the Dijkstra's algorithm n times, in this case 7 times, would help us get the shortest path from every source to every destination in this network. Effectively, the algorithm that we have seen actually solves all four of them, but with certain increasing amount of computation and complexity. Effectively, it solves both these in a single stroke, given 1 as source and 7 as destination. In a single pass of the algorithm, either through this method or through this method, we can find out the shortest distance from 1 to 7, which is from a given source to a given destination.

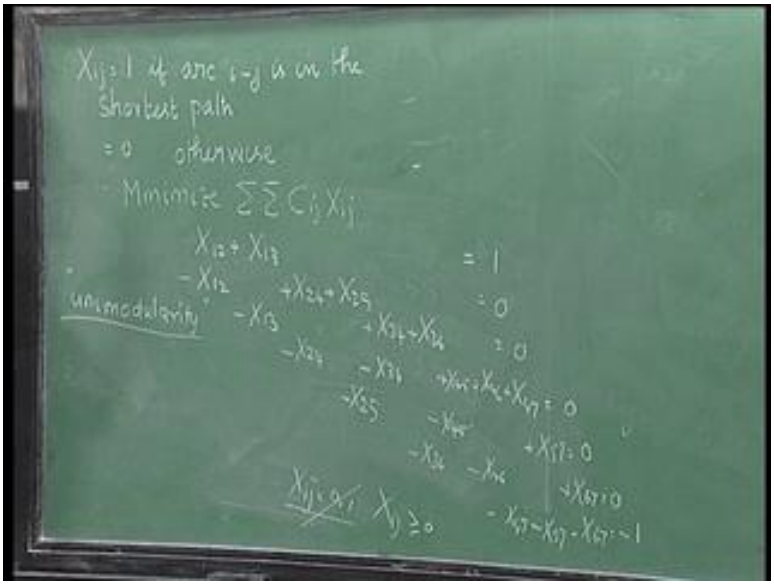
We can also find the shortest distance from a given source 1 to every other node as destination using the same algorithm. This is one more pass of the algorithm, which is a backward pass; whereas, this is a little more complex where we have to run this algorithm 7 times or (n minus 1) times, 6 times as the case may be. We can find out the shortest distance from every source node to every destination node on this network. There are other algorithms which try to handle this problem a little more efficiently where they specifically address this problem and where we find out the shortest distance from every node to every other node in the network.

There is a more popular algorithm called the Floyd's algorithm which we are not going to look at in this course. Nevertheless, we understand that we can run the Dijkstra's algorithm (n minus 1) times with each of these nodes as the starting nodes. Keeping each of these nodes as a starting node and solving this problem to find out the shortest path from a given source node to every destination node we will be able to handle the fourth one. Effectively, we understand that we can

use this algorithm to solve all these four problems though there are slightly more efficient algorithms to solve this.

With this, let us come back and try to understand how the Dijkstra's algorithm is optimal. Or how do we confidently say that, whatever we have done here or the labels essentially represent the correct shortest path and the corresponding shortest distance. In order to do that, we first formulate the shortest path problem as a linear programming problem and then try to write the dual of the linear programming problem to understand how these labels work.

(Refer Slide Time: 14:20)



Let us write the linear programming formulation for this. The formulation will be like this: X_{ij} equal to 1, if arc i - j is in the shortest path and is equal to 0, otherwise. The objective function will be to minimize $C_{ij}X_{ij}$, so X_{ij} will be equal to 1, if arc i - j is in the shortest path and 0 otherwise.

Now, if we take this node 1, which is a starting node, we should have at least one arc that leaves this node. We should write a constraint that X_{12} plus X_{13} should be equal to 1, because we have to leave this node in order to reach this destination. We can leave this node either through this or through this. We say X_{12} plus X_{13} is equal to 1. If we take this node, this is an intermediate node. The shortest path may pass through this, may not pass through this. This being an intermediate node the only thing we need to do is to model what is called the flow balance equation. If this

lies in the shortest path, then you enter and then you leave. If it does not lie in the shortest path you do not worry about it.

That is written in the form of a constraint which is like this: minus X_{12} plus X_{24} plus X_{25} is equal to 0. What does this constraint do if this lies in the shortest path? Then, you have to enter this node which means X_{12} should be equal to 1, which also means that you should leave this node by either to 4 or to 5, so the constraint will take care of it, X_{12} is 1. Then this contribution is minus 1, so one of them will take a plus 1, so that you get a 0. Therefore, if it enters 2 through X_{12}, it has to leave either through X_{24} or through X_{45}.

If it does not lie in the shortest path then this will be 0, automatically these two will be 0 and therefore, this constraint will take care of that requirement. For any intermediate node, all we need to do is to model the flow balance equation, so whatever comes in should go out. For this node, we will have minus X_{13} plus X_{34} plus X_{36} is equal to 0. For this 1, we will have minus X_{24} minus X_{34} plus X_{45} plus X_{46} plus X_{47} is equal to 0.

Remember, there are 5 arcs associated with this node. Two of them enter this node and three of them leave this node. For 5, there are three arcs associated, so you get minus X_{25} minus X_{45} plus X_{57} is equal to 0. For 6, we get minus X_{36} minus X_{46} plus X_{67} is equal to 0 and for 7, 7 being the destination node, you have to reach 7. So, we write X_{57} or X_{47} or X_{67}, sum of them should be equal to 1. So the constraint will be X_{57} plus X_{47} plus X_{67} is equal to 1. We write it in a slightly different way. We write it as minus X_{47} minus X_{57} minus X_{67} is equal to minus 1. Please note that this constraint should ordinarily read as X_{47} plus X_{57} plus X_{67} is equal to 1, but we have multiplied this by minus 1. We have written minus X_{47} minus X_{57} minus X_{67} is equal to minus 1.

We have X_{ij} is equal to 0 1. This completes the linear or linear integer programming formulation of the shortest path problem, because X_{ij} is equal to 0 1, makes X_{ij} a binary variable. Therefore this formulation is a linear integer formulation where all the variables are restricted to be binary variables.

Let us go back and try to understand that this problem is a network problem. We have seen some characteristics of network problems earlier when we have looked at transportation and assignment problems. We also said the transportation and assignment problems have an

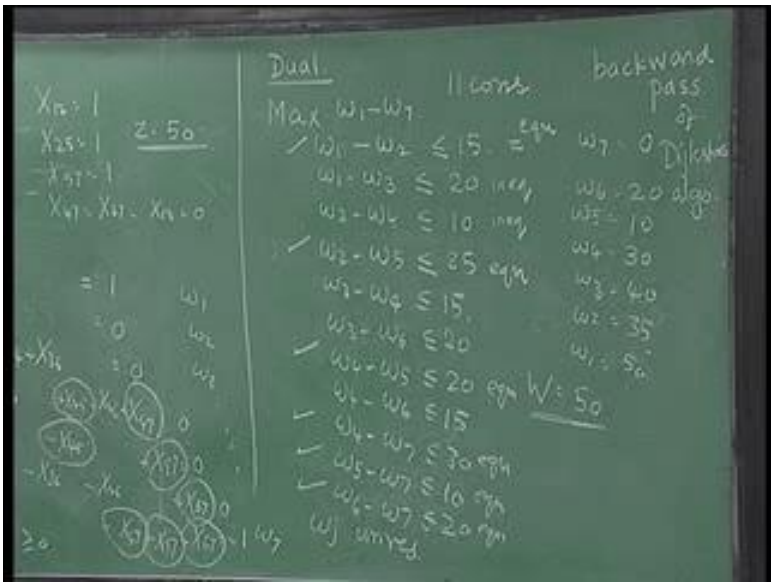
important property called unimodularity and because of this unimodularity property, even if you solve the linear integer problem by relaxing this binary variable and by writing it as continuous variable and by relaxing this we would still get solutions which are either 0 or 1.

We have already seen a little bit about this unimodularity property. Unimodularity property would simply mean the rank of the constraint coefficient matrix. We can actually show the way this matrix is structure simple addition on the left-hand side and simple addition on the right-hand side, would give us 0 equal to 0. In fact, one of the reasons we have written this in this form, particularly here, is to facilitate that, if you add all the left-hand sides and all the right-hand sides, you will see this 1 and minus 1 cancel out. Every X_{ij} will cancel out to give a 0 is equal to 0 which tells us that this is a linearly dependent system of equations.

Before that we also observe that this formulation will have as many constraints as the number of nodes or vertices and as many variables as a number of arcs or edges. There are n vertices, we have a linearly dependent system and we can also show that the rank is (n minus 1). Then, we also realise that the determinant of the corresponding matrix will have a plus 1 or a minus 1. Because of that property, B inverse will be integer value and therefore, the actual values of X_{ij} will only be integers. In this case 0 1, even if we relax the binary assumption and treat it as a continuous variables because of this unimodularity property, we can ignore this 0 1, treat it as a continuous variables, solve it as a linear programming problems. We would still get values for 0 1, for these variables.

Another way of showing unimodularity is that, if we have a constraint coefficient matrix, such that it can be written in this form, where every variable appears in only two constraints, whether plus 1 and minus 1, then we could show that, this one is unimodularity. In more than one way, we can establish the unimodularity characteristic of this matrix. Therefore, we can relax this 0 1 assumption and then treat it as a linear programming problem. Once we treated as a linear programming problem, it is easy to write the dual. We start writing the dual of this problem and therefore we introduce dual variables w_1 to w_7, call them w_1, w_2, w_3 and so on, till w_7.

(Refer Slide Time: 23:07)



Now, we write the dual of this problem. From duality, once we have defined dual variables w_1 to w_7, primal being a minimization problem, dual is a maximization problem. We will get maximize w_1 minus w_7, 1 into w_1 minus 1 into w_7; will be the objective function, maximize w_1 minus w_7. Each variable X_{ij} appears in the i^{th} and the j^{th} constraint here. We look at this w_1 minus w_2. If we look at variable X_{12} w_1 minuss w_2, we have a c_{12} here. Remember, the primal is a minimization problem; dual is a maximization problem. So minimization problem with all greater than or equal to constraint would give us some maximization problem with less than or equal to this. w_i minus w_j is less than or equal to C_{ij}. In our case, w_1 minus w_2 will be less than or equal to c_{12} which is 15.

Similarly, we can write w_1 minus w_3 is less than or equal to 20. From this, we now start writing w_2 minus w_4 is less than or equal to 10; w_2 minus w_5 is less than or equal to 25; w_3 minus w_4 is less than or equal to 15; w_3 minus w_6 is less than or equal to 20; w_4 minus w_5 is less than or equal to 20; w_4 minus w_6 is less than or equal to 15; w_4 minus w_7 less than or equal to 3;. w_5 minus w_7 less than or equal to 10; w_6 minus w_7 less than or equal to 20; and importantly, w_j is unrestricted in size. The unrestricted in size comes because all the primal constraints are equations, so dual variables are unrestricted in size. This completes the dual of the problem. If one can also see that the dual has seven variables corresponding to seven constraints of the primal, which correspond

to seven nodes in this network. The primal has twelve variables, which correspond to eleven arcs.

There are eleven arcs in this network. There are eleven variables here. These correspond to 11 constraints in the dual. Dual will have as many variables as the number of nodes and as many constraints as the number of arcs. Where, the primal will have as many constraints as a number of nodes and as many variables as the number of arcs. With this primal and with this dual, let us try and get a solution to this problem.

Let us start with w_7 equal to 0 that we like to put. So, w_7 is equal to 0, would actually give us from here w_6 is 20 and w_5 is 10 out of this. This would give us w_4 as less than or equal to 30. This would give us w_4 less than are equal to 35. This would give w_4 less than or equal to 30. We write w_4 is equal to 30. From this, using this 20, w_3 is less than equal to 40, so, w_3 is less than equal to 45. You get w_3 is 40. From here w_2 is 35, w_2 minus w_4 would give us 40, this would give us w_2 is equal to 35 and from here, we have w_1 is less than or equal to 60. w_1 is less than or equal to 50, so, w_1 is equal to 50.

If we start with the solution w_7 equal to 0 and work backwards, we get w_7 equal to 0, w_6 is 20, w_5 is 10, w_4 is 30, w_3 is 40, w_2 is 35, and w_1 is 50; w_1 minus w_7, so this w value is 50. We realise that this is a feasible solution to this dual. This is feasible to the dual in the sense that now if we look at this value w_1 minus w_2 is equal to 15, this is satisfied as an equation. w_1 minus w_3 is 50 minus 40 is less than, so this is an inequality; w_2 minus w_4 is 5, so inequality; w_2 minus w_5, so, equation; w_3 minus w_4 is 10 inequality; w_3 minus w_6, 20, inequality; w_4 minus w_5, 20, equation; w_4 minus w_6 inequality; w_4 minus w_7 equation; w_5 minus w_7 equation and w_6 minus w_7 equation.

Now, all these dual feasible solution satisfies all these unrestricted inside. We also realise that, this is satisfied as an equation, this is satisfied as an equation, this is satisfied as an equation, this is satisfied as an equation. If we apply complementary slackness to this dual feasible solution, wherever it is satisfied as an equation, the corresponding variable is a basic variable in the primal. X_{12} is a basic variable. I am just circling them so X_{12} is a basic variable. 2 5 is a basic variable. I am again doing this, now 4 5 is a basic variable. You may have 4 7 as a basic variable, I have 5 7 and I have 6 7. These are my basic variables.

There are six basic variables, which is also understandable. There are seven constraints. It is a linearly dependent system, so we will have (n minus 1) basic variable. There are six basic variables. Now, corresponding to these basic variables, can we define a solution to the primal? This would give us X_{12} is equal to 1; this is satisfied because X_{12} is 1, X_{12} plus X_{25} is equal to 0. Remember, these are non-basic and therefore they take value 0. Therefore minus X_{12} plus X_{25} is equal to 0 would give us X_{25} is equal to 1. As far as this is concerned, X_{13}, X_{34}, X_{36}. We go back to this 1 3, 1 is 50, 3 is 40 so inequality. 3 is 40, 30 inequality. 4 6 is 30 plus 20, again inequality. We come back to this, 4 5 and 4 7 right now, we put X_{25} is equal to 1.

This would give us a contribution of minus 1. We have four minus X_{45} and X_{57} is equal to 0. This would give us X_{57} is equal to 1. From here, you would have minus X_{57} is equal to minus 1. You will have X_{47} is equal to X_{67} equal to anyone from here. X_{13} or X_{34} or X_{36} is equal to 0. We would have one more basic variable to represent this equation. But then that basic variable would also take a value 0. So corresponding to this feasible solution, we observe that X_{12} is equal to 1, X_{25} is equal to 1, X_{57} is equal to 1, satisfies the complementary slackness condition. It also provides a basic feasible solution to the primal. This would give us Z is equal to 50, 1 to 2 is 15, 2 to 5 is 25, 5 to 7 is 10 so this would give us Z is equal to 50.

We realize two things: one is, having written the dual, and we have got a dual feasible solution by inspection. Corresponding to this dual feasible solution we applied complementary slackness. We first found out which are the ones that are satisfied as equation and which are the ones that are satisfied as inequalities. Those that are satisfied as equation, we go back here; we try to identify the corresponding basic variable for those that are satisfied as an equation. From that we observe that 1 to 2, 2 to 5, 5 to 7 is equal to 1 is primal feasible. In more than one way we can show the optimality. One is that, we have a primal a dual feasible solution. We apply complementary slackness and we obtained primal feasible solution.

Another way is to show that we have a dual feasible solution with value equal to 50; we have a primal feasible solution, with value equal to 50. They automatically, the values being equal they satisfy the complementary slackness condition, therefore they are optimum. This is a way to understand that we get the same 50 here, as the optimal solution, both from the primal and the dual of the linear programming problem. We have got the same 50 here, which represents the

optimal solution. This is a way to show from the linear programming formulation that the answer you have got here, which is 50, is indeed the optimal solution.

We need to discuss a little bit more on what do these w s represent? This w's effectively representing the labels that we have actually given here. There are seven values of w. We also realise that one of them is 0. Then we progressively move towards getting this 50. Alternately, if we look at this carefully, the seven labels, that we have given here for w_1 to w_7 are actually different from the labels that we obtained there. it is because we have w_1 is 0 here, w_1 is 50 there, w_2 is 15 here, w_2 35 there, w_3 is 20 here, w_3 was 40 there, w_4 is 25 here, w_4 was 30. You got different values of these labels.

Is there any relationship between these labels and these values that we actually got? Yes, there is a relationship between these values that we have here, as well as these labels and the relationship looks from this point of view, this is 0 and that was 50, from 7 it 50 and 0. Effectively, we add up to 50, which is the value of Z.

More importantly, the difference lies in the fact that, if we have done a backward pass of this algorithm, starting from 0 here, then the labels that we would have got would be exactly those labels that we have got here. The only difference is when I have described this part of the algorithm; I have described a forward pass starting from 0 for the source and 50 to the destination. When I describe the dual here, I started with destination equal to 0 and the source getting the value 50. Effectively, these labels represent the labels corresponding to a backward pass of the Dijkstra's algorithm, whereas here we have done the forward pass. We have already seen whether you do a forward pass or a backward pass, you will still get the shortest distance from the source 1 to the destination. The Dijkstra's algorithm is actually a very nice primal dual implementation of the linear programming formulation of the shortest path problem.

There is also an interesting relationship between these forward passes and the backward passes.

(Refer Slide Time: 37:57)

The dual was w_1 minus w_7 subject to w_i minus w_j is less than or equal to C_{ij}, then we said w_j unrestricted. What we had written here are the eleven constraints corresponding to the eleven arcs which satisfy w_i minus w_j is less than or equal to C_{ij}. We can do one more thing, because this is unrestricted. I can always replace w_j by minus w_j and therefore we will write this as maximize minus w_1 plus w_7, subject to minus w_i plus w_j is still less than or equal to C_{ij}. Then we say minus w_j or w_j unrestricted. This will lead us to a situation where you want to minimize w_7 minus w_1, subject to w minus w_i plus w_j is less than or equal to C_{ij} and w_j unrestricted.

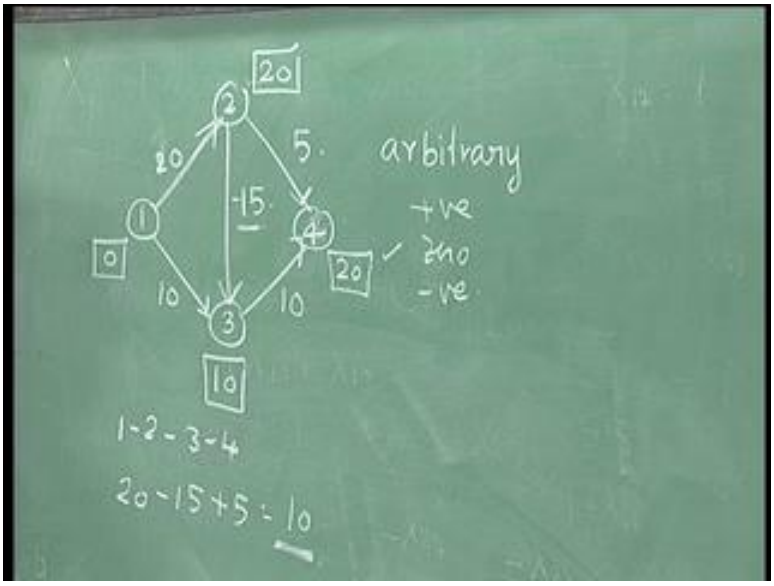
We start writing a dual in this form and then expand it, and then get the labels. Then, the corresponding labels that we would got here would exactly be these labels which are 0, 15, 20, etc respectively. There is a nice trick that lies in the fact that, because this w_j is unrestricted, we can simply replace w_j by a minus w_j and yet retain it.

So, starting now with w_7 equal to 0, would actually give us the forward pass of the Dijkstra's algorithm. The way we actually did it here we did the backward pass of the algorithm; whereas, in this place, we did the forward path of the algorithm. Nevertheless, we have seen a framework through which we will be able to show that the labels generated by the Dijkstra's algorithms were indeed optimal. That is the most important thing to learn from the shortest path point of view

The shortest path problem, even though formulated this way, because of unimodularity property has a continuous variable. We can write the dual and then we can get a dual solution by inspection and prove that the corresponding dual solution is indeed optimal. Provided, we generate the dual solution based on a certain principle and that principle was the same principle that we followed when we generated these labels. If each of these labels represent some dual variables and represents some w_j, then, what we did was, we made sure that w_i minus w_j is less than or equal to C_{ij}, then we then picked that w_j that was minimum.

If we are able do that then we are intuitively generating a feasible solution to the dual of the shortest path problem and by complementary slackness conditions, it will be optimal. Therefore, the labeling algorithm that we have seen for the Dijkstra's algorithm is indeed optimal. The next thing that can happen is we make one important assumption which was told in the earlier lecture. For the optimality of the Dijkstra's algorithm, all the C_{ij}s have to be greater than or equal to 0. We get into the next question, can sum of these C_{ij} be negative or can they take negative values? Let us see what happens if some of these C_{ij}s can take negative values.

(Refer Slide Time: 41:55)



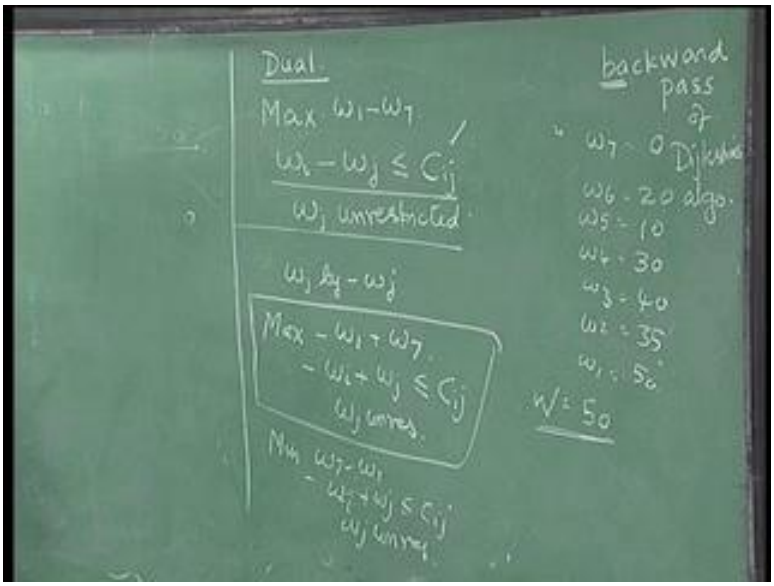
Let us take a simple network to explain this. Consider a simple network where the distance here is 20. We call this as 10, we call this as minus 15, call this as 10, and we call this as 5. Let these be the distances that we have here. If we start Dijkstra's algorithm for this, we will label this with

0. We have 1 to 2 is 20, 1 to 3 is 10, so we label this as 10. Then we will do this 1 to 2 is 20, 3 to 4 is 10 plus 10 is equal to 20. Let us say we label here as 20, now will do 10 plus 10 is 20, 20 plus 10, 25, so we would label this as 20 and say that the Dijkstra's algorithm terminates like this. The single pass of the Dijkstra's algorithm would give us a shortest distance from 1 to 4 as 20; whereas, we know that if we take the path 1 to 2, 2 to 3, 3 to 4 the distance is 20 minus 15 plus 5, which is actually 10 and not 20.

The reason we did not get the correct answer here is because we violated one of the assumptions that C_{ij} has to be greater than or equal to 0. We modify the shortest path algorithm, if we have these negatives. Such problems are called shortest path problems with arbitrary cost. Arbitrary cost would mean either a positive or a 0 or a negative cost. We can modify this algorithm through this.

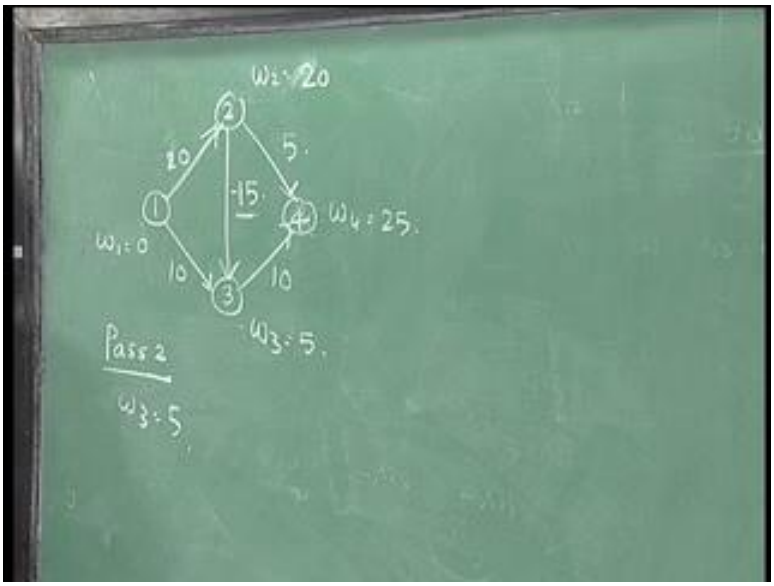
What we do there is this, instead of these labels, let me simply call them as w_1. One being the source, we call this as w_1 is equal to 0 will have w_2 equal to infinity, w_3 equal to infinity and w_4 is equal to infinity. What is the dual of the shortest path problem? Dual of the shortest path problem is w_i minus w_j is less than or equal to C_{ij}. So, first thing is if we are able to find w_i and w_j such that if we write the dual this way by changing this, we get w_j minus w_i is less than or equal to C_{ij}. If we replace this unrestricted variable by the negative of that, you will get w_j minus w_i is less than or equal to c_{ij} If you are able to get an ij such that w_j minus w_i is greater than C_{ij}, then we can actually correct the w_j. We look at this pair 1 2, w_j minus w_i is greater than C_{ij}.

(Refer Slide Time: 45:47)



Please observe that we are looking at this particular form of the dual. This is the first form of the dual. Here, we got w_i minus w_j less than equal to C_{ij}, w_j, unrestricted. This form comes because w_j is an unrestricted variable. I am replacing this w_j by minus w_j which would continue to be unrestricted. So, w_j unrestricted, but you will have minus i plus w_j is less than or equal to C_{ij}. We are using this form of the dual. Here, you have an ij pair such that w_j minus w_i is greater than $C_{ij,}$ so update this one. So, pass 1 of the algorithm. We have w_2 is updated to 20. Now again, go back to w_i minus w_j infinity and 0, w_j minus w_j is greater than C_{ij}, upgrade w_3 to 10. Here, you have w_j minus w_i is infinity minus 20 is greater than C_{ij}, upgrade w_4 to 25 such that you have w_j minus w_i is equal to C_{ij} in this case. Now, the values are updated from the original values till first pass you get 25.

(Refer Slide Time: 47:14)

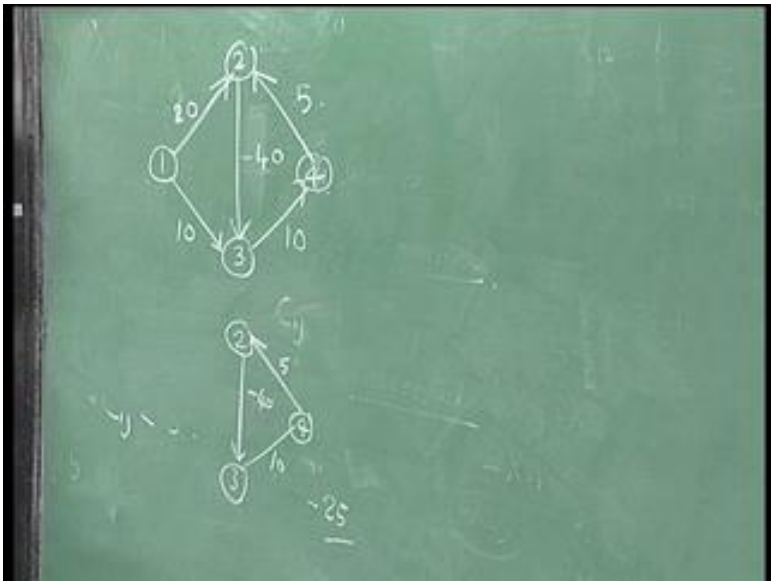


Go to the second pass and check whether for all ij, w_j minus w_j is less than or equal to C_{ij}. 20 minus 0 is equal to 20, satisfied. 10 minus 0 is equal to 10, satisfied. Come back to this w_j minus w_i is minus 10, is greater than C_{ij}, so, update this to 5, now this one is satisfied. Again in pass 3, verify for all ij, whether w_j minus w_i is less than or equal to C_{ij}. w_j minus w_i is equal to C_{ij}, w_j minus w_i is less than or equal to C_{ij} w_j is 5 minus 20 is less than or equal to 15, 25 minus 5 is less than or equal to 20. w_j minus w_i is greater than C_{ij}, 25 minus 5 is 20, 20 is greater than 10. Now change w_4 to 15, so w_4 becomes 15. Once again go back to the algorithm to check whether for all ij's, w_j minus w_j is less than or equal to C_{ij}, w_j minus w_j is equal to C_{ij}, w_j minus w_j is less than C_{ij}. w_j minus w_j is equal to C_{ij}, w_j minus w_j is less than C_{ij}, w_j minus w_j is equal to C_{ij}. So this set of w_js satisfies the condition: w_j minus w_j less than or equal to C_{ij}. Therefore, this value is optimal.

The only difference is if we have arbitrary cost, if we have some C_{ij} less than 0. It is necessary to run or modify the Dijkstra's algorithm, so that the dual condition is satisfied by the resultant once. The first type of thing which is a Dijkstra's algorithm, they come under the category of what is called label setting algorithm where you have C_{ij} greater than or equal to 0. Once the label has set, the label will not change. So these come under the category of label setting algorithms.

The optimality principle is that, once a label is fixed, which means I have determined the shortest path from the source to that node, I will not change the label. But when we have arbitrary cost like a negative coming here, then it may be necessary to correct the label as we move along and we apply this duality condition. These algorithms come under the category of order called label correcting algorithms. The labels that are set here 0, 20, etc., are now corrected. They are not fixed as it was done in the earlier case. So, they come under the category of order called label correcting algorithms. This way we can handle shortest path problems, even if some of them have arbitrary cost. There is another important thing that can happen.

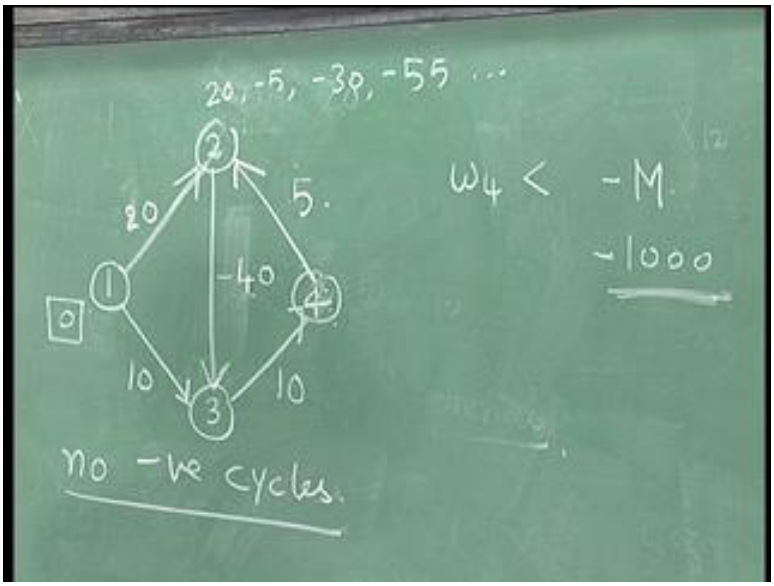
(Refer Slide Time: 51:17)



Let us look at this same network with the slight change. Let us say this cost is minus 40. Once again, it is very similar to the earlier example, except that one C_{ij} is negative and it is minus 40. Most importantly, it is going to create a big problem for us. Let us make another interesting change here. Instead of this 5, let me put this 5 and this arc here.

Let us look at a network like this. Remember the changes that I have made. One change is this is minus 40; here. I have changed the direction and it goes like this. The disadvantage with a network like this is, if we look at 2 to 3, 3 to 4, 4 to 2, we have a minus 40, 10 and 5. If we look at this cycle, this cycle has a total length of minus 25, minus 40 plus 10 minus 5. So, this has a total length of minus 25. This is a negative length cycle.

(Refer Slide Time: 52:53)



When we have a negative length cycle and we are interested in finding the shortest path from 1 to 4, 1 will always say that I can start with 0. I come back to 20, I complete one cycle. I get 20 minus 25, which is minus 5. I complete another cycle, I get minus 30, I get minus 55 and it goes on. The algorithm will not terminate here. One of the things is whether you use the basic Dijkstra's algorithm; we ensure that all C_{ij}'s are greater than or equal to 0.

There is no question of negative sign whereas in the modified label correcting algorithm, we said if we have arbitrary cost, we can still handle it, provided, there are no negative cycles. If there is a negative cycle in the network, then the label correcting algorithm will keep on correcting the labels till it starts getting negative values and it gets smaller and smaller, wj's become much smaller and smaller as we proceed.

The other question that comes is how do I know that there is a negative cycle in a network? This is a very small network. This has been created only to explain this principle. By inspection, one can say that this is a negative sign, but if we are dealing with a very large network, how would somebody know that there is a negative cycle here? The answer is, if we apply the label correcting algorithm and then we realize that these labels are becoming smaller and smaller, we fix a particular value like say, if w_4 happens to be less than equal to some minus big M or minus 1000 or some large value. Look at some arbitrarily defined large value and when one of the dual

variables becomes less than such a value, then the algorithm will terminate saying that there is a negative length cycle.

The label correcting algorithm does two things. One is, if there are arbitrary costs and there are no negative length cycles, then the algorithm will terminate by giving the optimum value. If there are negative length cycles, then the algorithm will terminate by saying that there is a negative length cycle. This is because somewhere, as we move along in this, we would started with 0, 20 so we would start with 20. After one cycle, we will get minus 5 because the length of this cycle is minus 25. After another cycle, we get minus 30 minus 55 and so on. Somewhere, the value keeps reducing and it will go to less than this 1000 and therefore the algorithm will terminate at that point saying there is a negative length cycle.

The algorithm will not find the shortest path from 1 to M. What we have effectively done is, we have now shown, first of all, the primary dual relationship that is associated with the Dijkstra's algorithm. We formulated the problem as a linear programming problem. We wrote the dual of that problem. Then, we derived the optimal solution to this problem using some ideas from duality. We first created a dual feasible solution here, based on an idea that we kept this as small as possible, so that we finally got to this 50. Then, we showed that applying the complementary slotness conditions to this dual feasible solution, we were able to get a primary feasible solution which was optimal. We also showed that there is relationship between both of them, while this represented the forward pass that represented backward pass.

We also said the underlying idea is to keep w_j minus w_j less than or equal to C_{ij} and then try to keep it as small as possible; small as possible comes by labeling here. The moment we know this as 15, we labeled it as the smallest possible value of 15. We labeled this or the smallest possible value of 25 such that w_j minus w_j is less than or equal to C_{ij}. We also showed why this is true by redefining the dual this way, so that we actually end up minimizing this, satisfying this condition. Then, we extended this idea to look at shortest path problems with arbitrary cost, some places where you have some negative values. Then, we modified that algorithm based on this duality procedure. We also had showed a way by which, if we have negative length cycle the algorithm will terminate without giving the optimal solution.

We will continue in a subsequent class on order called successive shortest path problem.