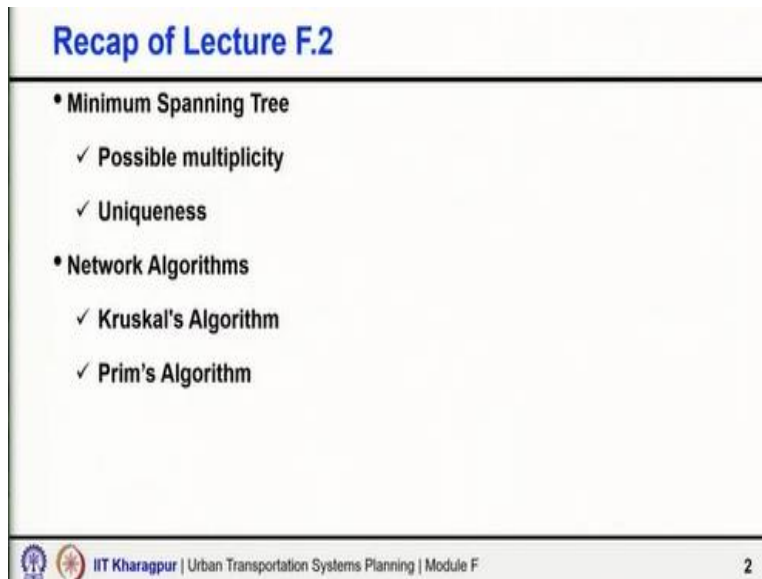


**Urban Transportation Systems Planning**  
**Prof. Bhargab Maitra**  
**Department of Civil Engineering**  
**Indian Institute of Technology, Kharagpur**

**Lecture - 43**  
**Network Algorithms-II**

Welcome to module F lecture 3. In the previous lecture we told you how the road network normally gets developed starting from village area with one or two really good roads all weather you know blacktop or concrete road.

**(Refer Slide Time: 00:32)**



The slide is titled "Recap of Lecture F.2" in blue text. It contains a bulleted list of topics covered in the lecture. The first bullet is "Minimum Spanning Tree", which has two sub-bullets: "Possible multiplicity" and "Uniqueness". The second bullet is "Network Algorithms", which has two sub-bullets: "Kruskal's Algorithm" and "Prim's Algorithm". At the bottom of the slide, there is a footer with the IIT Kharagpur logo and the text "IIT Kharagpur | Urban Transportation Systems Planning | Module F" and the number "2".

And then slowly targeting at least to get all the road connectivity to ensure you know connectivity of all the nodes that the first or the level two that is the first target rather. And then I said that with more and more population living in urban area more travel demand people start developing more and more roads to give direct connectivity with shorter travel time or shorter travel distance.

Sometimes to decongest some of the existing roads built another road. So, finally in fully developed network you have multiple path. And in the context of the first target that when you know we are trying to get at least what is the minimum road length that we should have. So, that we ensure connectivity of all the nodes in that context I told about the minimum spanning tree

discussed about the two properties possible multiplicity also the uniqueness when every link the cost is distinctly different.

And then we also discussed about two algorithms Kruskal's algorithm and Prim's algorithm which may be used to obtain the minimum spanning tree. With examples, we you know discussed and explained to you how stepwise or step by step how we find out finally the minimum spanning tree. But then that was like stage two development, where we just have basic minimum road to ensure connectivity of all the nodes.

From that point onwards with more and more population with more urbanization we start building more and more link additional links. And every time we are building a new road or new connectivity, we are doing we are creating loops alternative path for to one or more for one or more O-D pairs. So, like that we keep developing the root network. And finally, in a fully developed urban area the problem is very different.

There we have generally multiple paths or alternative paths available if we want to travel from one origin to another destination. And such kind of alternative paths are generally available for majority of the O-D pairs. So, in that case as you know normally everybody tries to travel by the shortest path. So, if can travel in 20 minutes using a path naturally, I will not select a path where it will take 40 minute or 30 minutes. So, for everybody it is so all of us we would like to travel using the shortest path or in the fastest possible time.

**(Refer Slide Time: 04:20)**

## Network Algorithms

### Shortest path

- Finding a path between two vertices (or nodes) such that the **sum of the weights of its constituent edges is minimized**
- Formally, given a weighted graph (that is, a set  $V$  of vertices, a set  $E$  of edges, and a real-valued weight function  $f : E \rightarrow \mathbb{R}$ ), and one element  $v$  of  $V$ , find a path  $P$  from  $v$  to a  $v'$  of  $V$  so that  $\sum f(p), p \in P$ , is minimal among all paths connecting  $v$  to  $v'$



So, in that background now we need to understand that what is the shortest path and how we can calculate the shortest path? So, in the context of shortest path what we are trying to find out a path between two vertices or two nodes. Such that some of the weights of its constituent edges is minimum sequence of links I can follow some sequence of link or another sequence of link to reach to the same destination.

Like that there are many alternative paths I am only selecting that path where some of the weights of its constituent edges is minimum among the alternative paths. Formally given a weighted graph and one element  $V$  of capital  $V$  that means one node of all the vertices or all the links all the nodes that are there in the network. Find a path from one destination to another destination.

So, that within this pair the total travel time or the travel cost is minimized, again cost and time are used interchangeably. Time is also a cost may be the direct cost of travel length may be direct you know vehicle operating cost it generally represents, what you are using as deterrence. If you use cost you are minimizing cost if you are using time then you are minimizing time whatever is the deterrence that you are using. That we are optimizing we are minimizing by selection of the path which is the shortest one.

**(Refer Slide Time: 06:34)**

## Network Algorithms

### Dijkstra's Algorithm

- Dijkstra's algorithm finds the **shortest path from the origin 'r' to every other node** when every link has a non-negative cost
- Furthermore, at each step it finds the shortest path to at least one additional node
- Prim's algorithm finds the minimum spanning tree (MST), while Dijkstra algorithm finds shortest path tree (SPT) from the given source



Likes we said two algorithms for minimum spanning to here also we are going to discuss two algorithms for find out the shortest path. Both are very popular algorithms. The first one is Dijkstra's algorithms. I shall little bit of course describe it in this form but it will be easy for you to understand and for me also it is easy to explain with an example. So, we shall definitely spend more time there to explain you.

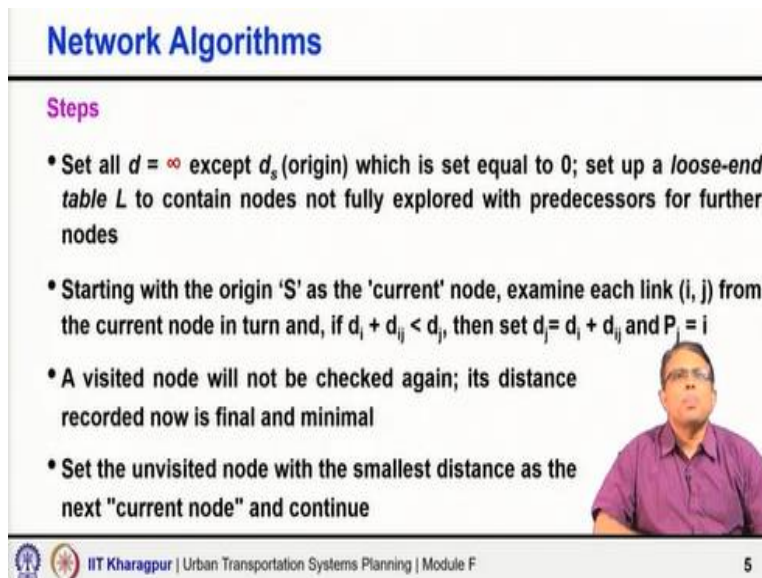
But let us try to give you some overall idea first before we take an example and then explain explaining the steps. In Dijkstra's algorithm we try to find out shortest path from one origin to all other destination node. When every link has non-negative cost so one very fundamental assumption and of course it is a very realistic assumption the cost is non negative how the time length or cost can be negative in this sense.

In this you know traffic assignment context network context in this context you know it can never be negative. So, what we get by the time we complete one set of iterations of this algorithm apply it. We get from one home node that is my starting node to all other nodes in the network what is the shortest path that we get? Now remember that many times many times I have seen students have confusion or they get confused between Dijkstra's algorithms they do lot of confusion between Dijkstra's algorithms and Prim's algorithm.

Prim's algorithm is finding giving me the minimum spanning tree. That means what is the network or a network that will ensure connectivity to all nodes n nodes with exactly n-1 link and with total minimum length or total minimum cost. Whereas Dijkstra's algorithms we are trying to get shortest path tree or you can call it SPT or shortest path from one given node to all other nodes in the network.

So, what from if I am starting node is 1 then 1 to 2, 1 to 3, 1 to 4, 1 to 5 for every destination what is the shortest path and what is the sequence of links that we need to select to reach to that particular destination as per the shortest path, that we will get.

(Refer Slide Time: 10:03)



**Network Algorithms**

**Steps**

- Set all  $d = \infty$  except  $d_s$  (origin) which is set equal to 0; set up a *loose-end table L* to contain nodes not fully explored with predecessors for further nodes
- Starting with the origin 'S' as the 'current' node, examine each link (i, j) from the current node in turn and, if  $d_i + d_{ij} < d_j$ , then set  $d_j = d_i + d_{ij}$  and  $P_j = i$
- A visited node will not be checked again; its distance recorded now is final and minimal
- Set the unvisited node with the smallest distance as the next "current node" and continue

IIT Kharagpur | Urban Transportation Systems Planning | Module F

5

Steps, first we assume that two types of level we will do here. Let us understand it in this manner. One is a temporary level another is a permanent level. So, at any stage when I am applying this algorithm every node will have a level either temporary or permanent. So, as we move stepwise iteration wise then you will see that in every iteration one node we shall level permanently.

So, once the permanent level is done for a node, we are not going to change anything there. Because that is why it is permanent so once it is level permanent that means we are not going to change it anymore. Other nodes tentative level will be there. Then tentative levels will keep changing from one iteration to another iteration. That means still we have not finalized. So, they

will keep changing but every iteration one node out of those temporary levels we shall make it a permanent level.

So, if I have got generally speaking  $n$  number of destinations, I have to find the shortest path then I will require any iteration. Every iteration one node will be level permanently. So, initially what we are assuming that from my starting node to all other node the cost is infinity or very large value infinity or very large value. So, my cost is a very large value and except for the starting node because I am starting with a node.

So, that is my origin node. So, there the cost will be 0 and I know the predecessor of that node is also that particular origin node. For all other nodes we will assume the cost as infinity and you can even assume the predecessor as the starting node. Sometimes we say that explicitly sometimes we may not say that explicitly, but the cost is taken as infinity or very high value. Now starting with that node, we explore now more and more possibilities in every iteration.

And wherever we will find that by changing the predecessor what is assigned till the previous or in the previous iteration, that is the update. If I am doing  $n$ th you know any  $j$ th iteration or any  $k$ th iteration then  $k - 1$ th iteration. Whatever was the temporary levels those are the best solutions so far. See if I find by changing the predecessor what I am discovering now in this iteration if I by changing the predecessor if I can reduce that cost further. Then I will change the predecessor and update the cost.

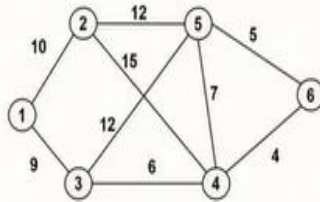
Otherwise my temporary level will remain as it was there in the previous iteration. And end of this iteration with all the finally updated temporary levels wherever the cost will be minimum I will that node I will transfer from temporary level to permanent level. That is what it is. So, let us take an example.

**(Refer Slide Time: 14:16)**

## Network Algorithms

### Example

Find out shortest path from node 1 to all the other nodes in this network using Dijkstra's algorithm



It is to go each or to go step by step. I have taken an example, where we have six nodes 1, 2, 3, 4, 5, 6 and the links are given as per the existing connectivity say 1 to 2, 1 to 3, 2 to 4, 2 to 5, 3 to 5, 3 to 4, 4 to 2 is there already 3 is already there 4 to 5, 4 to 6 and 5 is connected to 6. That those are the connections which are existing. Other connections are not there and, in each case, what is the cost that is known.

So, let us say now we want to find out shortest path from node one and as I said what distance will give from one home node to all other home node. So, we eventually want to find out using Dijkstra's algorithms in one set of iterations the shortest path from one to all other nodes. That means 1 to 2, 1 to 3, 1 to 4, 1 to 5, 1 to 6 stepwise. So, what I said first, first step is you can say first step you can say 0th step it is all you know the way you want to define it.

The very first thing assumption is that except for one if one is my starting node then for one the predecessor is definitely one because that is my starting node. And the cost is zero. Because that is my starting node for all others in the beginning, I have not even looked at the network but I will assume the cost is infinity. And you can also assume that the predecessor is that node or one or whatever it is.

Predecessor is that node itself you can consider our predecessor is one. So, let us say predecessor is one the home node and the cost is infinity. Here the predecessor is really not so important but

main consideration is the cost. Eventually we will assign the predecessor also because that is the way the links will get explored or you know connection will get explored.

**(Refer Slide Time: 16:56)**

**Network Algorithms**

Current Node	V	Pr	d
1	1	1	0
	2	-	∞
	3	-	∞
	4	-	∞
	5	-	∞
	6	-	∞

V- Node  
Pr- Predecessor  
d- distance

**Previously**

**Final sptSet**

Node	Predecessor	Distance
1	1	0

IIT Kharagpur | Urban Transportation Systems Planning | Module F 7

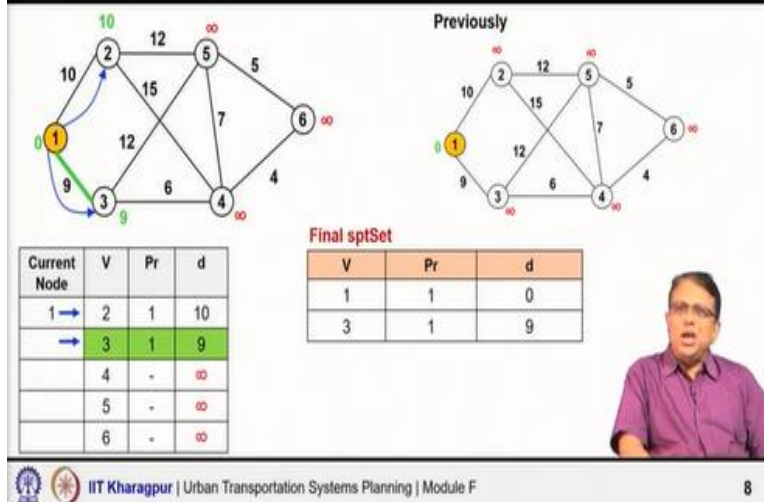
So, that is what I am showing here. My step 0 or step one whatever you say for one node you see that in blue, it is written as zero because that is the total cost it is the starting point. And to 2, 3, 4, 5, 6 the cost is infinity. Because I still do not know which path, I am going to take what are the cost nothing is known. I am not even looked at the network, I have only identified my starting node and I know starting node the predecessor is that cost is zero for all others it is infinity.

So, nothing you know you can follow the table 1 to 1 0. So, in a way I can only permanently level my starting node which is node number one here, with predecessor one starting node because that is the starting node and cost is zero that is where I am starting. Now one is established with zero cost and one as the predecessor node now my next iteration.

**(Refer Slide Time: 18:25)**



## Network Algorithms



Now one is established so now I am looking at the matrix or looking at the connectivity here for the classroom example or when I am explaining we will look at the graph. But actually, the algorithm will work it will take a matrix and as per that it will start doing the calculation and explore various possibilities. So, 1 is connected to 2 because that is the connection which is existing and I know from 1 I have reached 1 so I know 1 to 2 exist 1 to 3 exist.

Let us say 1 to 2 now I know that 1 to 2 the cost is how much 10 and what I assumed as the cost infinity in the beginning. Predecessor as I said is not so important you can but whatever it is you can consider the predecessor as 1 starting node. But now I have find out an alternative path maybe even if you take the predecessor as 1 it is still now 1 but I have found out in another part rather than my assumed cost I have actually discovered a path, where the cost is 10.

So, if I travel to 2 from 1, I have a connection and where the cost is 10 and 10 is lesser than infinity. So, what I need to do I will update my cost as 10 instead of infinity. So, that is what is written in green infinity is taken out because 10 is lesser than infinity so I have got 10. And I made my predecessor now as 1 for this node 2. So, this blue arrow in the table below the graph here the connections are shown.

You can see so 1 is connected to 2 and similarly 1 is connected to also 3 and the cost is 9, now 9 is again less than infinity. So, I will connect I will update the cost and with predecessor as 1 so I

know that I can come from 1 to 3 with a cost 9 which is lesser than the infinity. The remaining 4, 5 and 6 I have no information till now, I do not know. So, my cost still remains infinity and that is what is my present condition of tentative levels.

I only explore two connections, 1 to 2, 1 to 3 both cases I found that with one as predecessor if I connected directly 1 to 2 then my cost is lesser so I have updated. Now I am not writing the for vertices 2 the cost is infinity but I am writing the cost as 10 for 3 node 3 I am writing now the cost as 9 not infinity and I have written the predecessor correctly. For all others I have not indicated the predecessor if you want you could still keep it.

But the cost is infinity. Now I said that every iteration I will permanently level one node. So, now I look at this distance or the cost whatever cost distance whatever you say. Now I look at this list I find my values are 9, 10 and remaining are infinity. So, among all these values what is the least one? 9, that is for what for what is on node 3. So, 3 I am permanently leveling I am transferring three from tentative levels to permanent level, so, you can see this final SPT set.

Now three is transferred here. Earlier I had only one node, node number 1 with predecessor one cost 0. Now I have permanently leveled also in this iteration node number 3 with predecessor 1 and cost has 9. So, one will node three will get transferred from temporary level to permanent level it will not three will not go back again to that temporary level, because we are not going to touch it anymore.

Now one pertinent question could be in your mind that how we can be so confident that three we can no way reach through any other possibilities with a lesser cost what could be the reason. The reason is very simple. We said one assumption the cost is non-negative either it is zero or some positive value in all practical case it will always be a positive value. So, look at this thing I know 1 is connected to 2, 1 is connected to 3 maybe 2 is further connected to some other node which may eventually may get connected to some other node.

And I may always explore some path to come back to my node 3. But then can the cost be lesser than 9? No. It cannot it is not possible, why? Because any I want to through 2 if I want to explore

anything the cost will always be 10 plus something and I already know that nine is even lesser than ten. So, any path through two will always have cost 10 plus 10 or 10 plus. So, I am saying there cannot be any better path cheaper path, so just permanently level three.

(Refer Slide Time: 25:17)

**Network Algorithms**

Previously

Current Node	V	Pr	d
3	2	1	10
→	4	3	15
→	5	3	21
	6	-	∞

Final sptSet		
V	Pr	d
1	1	0
3	1	9
2	1	10

IIT Kharagpur | Urban Transportation Systems Planning | Module F 9

Once I have done three now my new connections are there. Now I discovered I still have 0 to 1 to 2 in the temporary level with you know 10 cost. But now three is also discovered with cost 9 now 3 is connected to 4 connected to 5. Now if I want to go to 4 using this part 1, 3, 4 then my total cost is  $9 + 6$ , 15 is less than infinity 15 is less than infinity. So, I will update the cost of 5 from infinity to 15.

That is what I have done. And also change the predecessor of node 4 as 3. That is what I have done. So, that means I am saying 4 predecessors of 4 is 3 predecessors of 3 is 1. So, you can always stress back you know the path what it is going to follow. So, I have updated the cost for node 4, similarly 5 if I want to go 1 to 3 to 5, then  $9 + 6$ ,  $9 + 12$  21, 21 is lesser than infinity. So, I will change the permanent temporary level.

So, for 5 the predecessor becomes 3 cos 21 for 6 I have no information till now so my temporary level as per my temporary level my costs are 10 50 21 and infinity which is the cheapest 10 for which vertex are vertices or which node 2. So, 2 is now permanently leveled. So, I have shifted 2

to the final speed set with predecessor 1 cost as 10. So, remaining I have 4 with 15 cost 5 with 21 cost 6 with infinity cost. Now let us go to the next step.

**(Refer Slide Time: 27:47)**

**Network Algorithms**

Previously

Current Node	V	Pr	d
2 →	4	2	25
→	5	2	22
	4	3	15
	5	3	21
	6	-	∞

Final sptSet		
V	Pr	d
1	1	0
3	1	9
2	1	10
4	3	15

IIT Kharagpur | Urban Transportation Systems Planning | Module F 10

Now 4 is also permanently level. So, I have explored more I know that 4 is also connected to 6, 4 is so 4 if I consider so what we did 2 and 3, 4 we have not yet taken. So, the next connections so far, I have permanently leveled 1, 2, and 3. So, 2 is permanently level so from two I have now again explored 2 to 5 and 2 to 4 now if I go to 5 from 2 my cost is  $10 + 12 = 22$ . And 5 it is  $10 + 12 = 22$ .

And if I go what was my earlier cost my earlier cost was 21 from 3. So, 21 is lesser than 22 so I do not update I do not update. So, although I am exploring new connections I do not update. So, I have written the (( ))(29:28)) with 2 as the new node permanent level node I have explored this connection 4 to with predecessor as 2, but cost is 25 and for vertices 5 or node 5 with predecessor, so, it is 22.

But in both cases for node 5 if you take earlier it was 21 now it is 22 so I do not update for 4 earlier it was 15, now it is 25 again 25 is higher than 15 so I do not update. So, I do not change the predecessor I do not update the cost my temporary level 4, 4, 5 remains same. So, now my levels are 4, 5, 6 four is having the minimum cost 15, so I now permanently level four and transfer 4 there. So, 1 3 2 4 permanently leveled.

(Refer Slide Time: 30:40)

### Network Algorithms

10 12 15+7=22>21  
10 15 5  
9 12 7  
9 6 4  
9 15

Previously

10 12 15+12=22 > 21  
10 15 5  
9 12 7  
9 6 4  
9 15+25 > 15

Current Node	V	Pr	d
4	5	3	21
→	<del>5</del>	<del>4</del>	<del>22</del>
→	6	4	19

Final sptSet

V	Pr	d
1	1	0
3	1	9
2	1	10
4	3	15
6	4	19

IIT Kharagpur | Urban Transportation Systems Planning | Module F

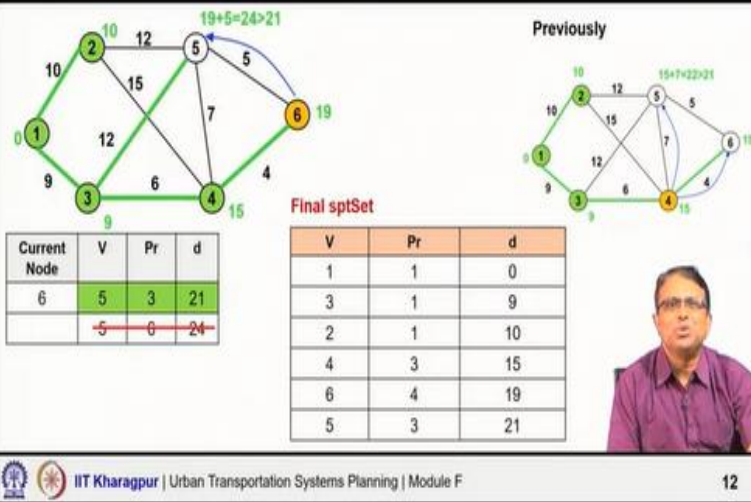
11

Next once you have level 4 my new connection here is 4 to 6 and 4 to 5. So, 6 again get updated because 6 was infinity so far. So, now I find if I take 1, 3, 4, 6, then  $9 + 6 + 4 = 19$ , 19 is lesser than infinity. So, I take it and with four if I go to five then my cost will become  $9 + 6 + 15 + 7 = 22$ . But 22 is less than is higher than 21 already the other path which is there till now, that means 1, 2, 3 to 5 so do not update that so I have deleted that temporary thing.

We considered but we did not update just delete it. So, my update is now 5 and 6 these are by 2 nodes 5 case the predecessor is 3 cost is 21, 6 predecessors is 4 cost is 9. So, I now permanently level 6.

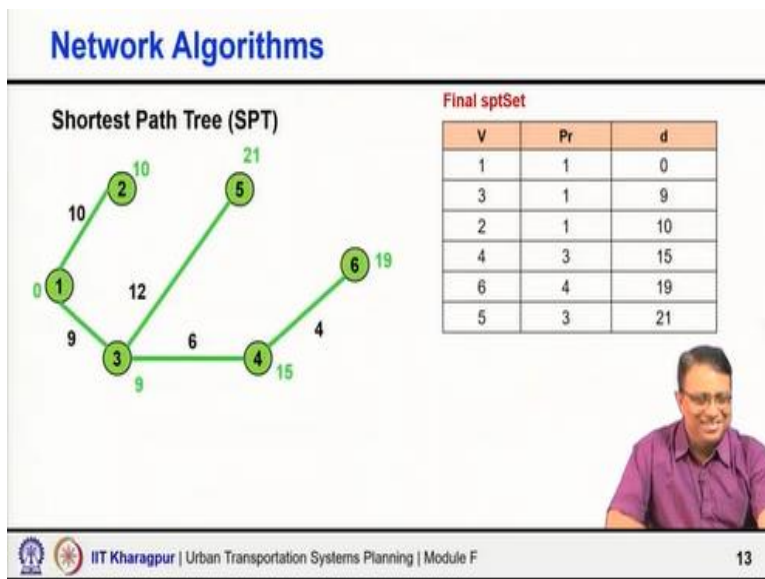
(Refer Slide Time: 31:56)

## Network Algorithms



And then next only the five will be remaining, so five also comes. So, that is the way. Now you have shortest path from one home node that is in this case one to all other home node 2, 3, 4, 5, 6 each case you know the predecessor. Then that node says for I know 1, 3, 4, 6 predecessors of 6 is 4 predecessors of 4 is 3 predecessors of 3 is 1. So, I know the path 1, 3, 4, 6 and what is the cost that you get from this table.

(Refer Slide Time: 32:32)



So, that is what is my shortest path, how it is going and side table tells me what is actually the cost.

(Refer Slide Time: 32:42)

## Summary

- Shortest Path
- Dijkstra's Algorithm
  - ✓ Shortest Path Tree

So, in this course in this particular lecture we discussed about told you about the context of shortest path why it is important. And one algorithm Dijkstra's algorithm that we said and we have shown you with an example how Dijkstra's algorithm can be applied to get a solution. So, with this I close this course this lecture and we shall continue in the next lecture one more algorithm I will discuss that is called Floyd's algorithm. Thank you so much.