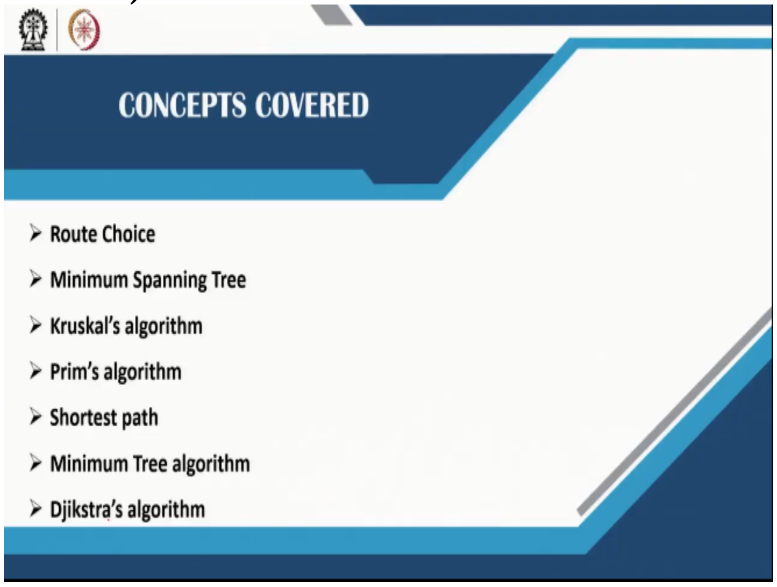
Urban Landuse and Transportation Planning

Prof. Debapratim Pandit
Department of Architecture and Regional Planning
Indian Institute of Technology, Kharagpur

Lecture - 42
Route Choice

(Refer Slide Time: 00:28)

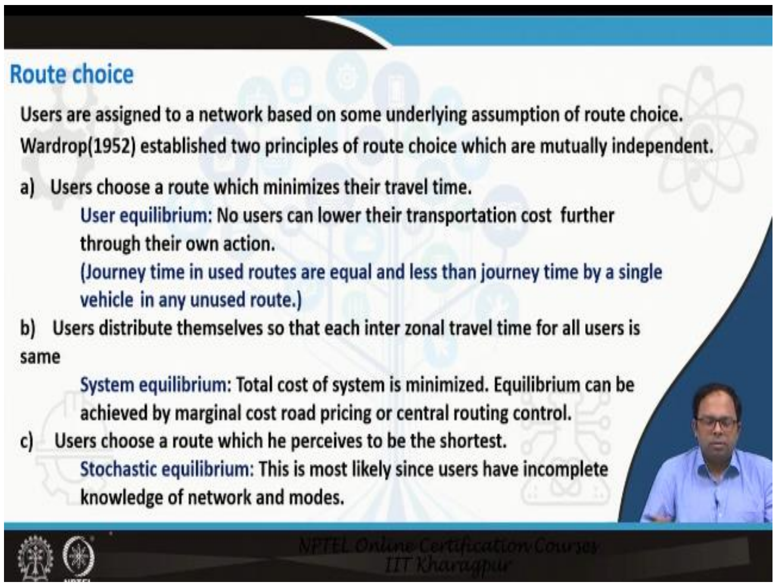


CONCEPTS COVERED

- Route Choice
- Minimum Spanning Tree
- Kruskal's algorithm
- Prim's algorithm
- Shortest path
- Minimum Tree algorithm
- Djikstra's algorithm

In lecture 42 we will talk about route choice. In this lecture, route choice will be discussed in the context of trip assignment. We will also cover different algorithms utilised to determine attractive routes. Both minimum spanning tree and shortest path algorithms will be discussed. We will solve examples for minimum spanning tree using Kruskal's algorithm and Prim's algorithm. Finally, shortest path algorithms like minimum tree algorithm and Djikstra's algorithm will also be covered in this lecture.

(Refer Slide Time: 00:57)


Route choice

Users are assigned to a network based on some underlying assumption of route choice.
Wardrop(1952) established two principles of route choice which are mutually independent.

a) Users choose a route which minimizes their travel time.
User equilibrium: No users can lower their transportation cost further through their own action.
(Journey time in used routes are equal and less than journey time by a single vehicle in any unused route.)

b) Users distribute themselves so that each inter zonal travel time for all users is same
System equilibrium: Total cost of system is minimized. Equilibrium can be achieved by marginal cost road pricing or central routing control.

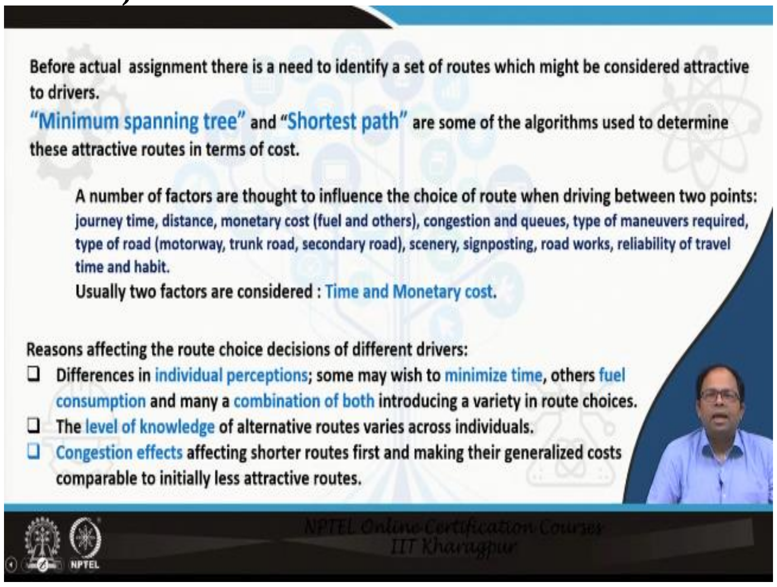
c) Users choose a route which he perceives to be the shortest.
Stochastic equilibrium: This is most likely since users have incomplete knowledge of network and modes.

NPTEL Online Certification Courses
IIT Kharagpur

Route Choice

Link assignment can be conducted after the determination of the route chosen by the user to travel between his origin and destination. Therefore, the users are assigned to a network based on some underlying assumptions based on route choice. These assumptions are based on the principles postulated by Wardrop (1952). Wardrop established two principles of route choice. Both the principles are mutually independent. He suggested two forms of equilibriums, *user equilibrium* and *system equilibrium*. The user equilibrium is based on the assumption that the users choose a route which minimises their travel time. The equilibrium for such cases is said to be attained when no user will be able to lower their transportation costs further by changing their action. The journey time has a specific characteristic for user equilibrium. The journey time in the used routes between the different origin and destination pairs will be equal. Moreover, the journey time in used routes will always be less than the time taken by a single vehicle in any unused routes. Therefore, the cumulative journey time of the unused routes will be higher than the cumulative journey time in used routes. The first Wardrop principle is based on the concept that the route with minimum travel time will be chosen. So, automatically the used routes are those which have minimum travel time according to users. Any other routes which are not chosen will have higher travel time compared to chosen route. The system equilibrium is based on the second principle of Wardrop. This principle is based on the assumption that the users distribute themselves so that each inter zonal travel time for all users is same. Under this principle, the equilibrium can be achieved if the total cost of system is minimised. For example, system equilibrium can be attained with road pricing or central routing control. Such equilibrium can be established when the city authority tries to provide equal impedance through the introduction of road pricing schemes or control of the traffic signals at the central level. But system equilibrium is difficult to achieve since it does not consider the behavioural principles. It is difficult to optimise a system since individuals choose routes based on their perception of travel time, preference and. information. Therefore, user equilibrium is commonly used in traffic assignment. But users do not have the perfect information regarding the network. So, another variant of the user equilibrium is introduced. The equilibrium where users choose a route which he perceives to be the shortest is also known as stochastic equilibrium. The main assumption of this equilibrium is that the users have incomplete knowledge of networks and transportation modes. In this equilibrium, all the users have routes either equal to or greater than their perceived costs. The main difference between user and stochastic equilibrium is the use of travel time in both the models. While the stochastic equilibrium models use variability of perceived route time, user equilibrium models use a single definition for all the users.

(Refer Slide Time: 05:02)



Before actual assignment there is a need to identify a set of routes which might be considered attractive to drivers.

"Minimum spanning tree" and "Shortest path" are some of the algorithms used to determine these attractive routes in terms of cost.

A number of factors are thought to influence the choice of route when driving between two points: journey time, distance, monetary cost (fuel and others), congestion and queues, type of maneuvers required, type of road (motorway, trunk road, secondary road), scenery, signposting, road works, reliability of travel time and habit.

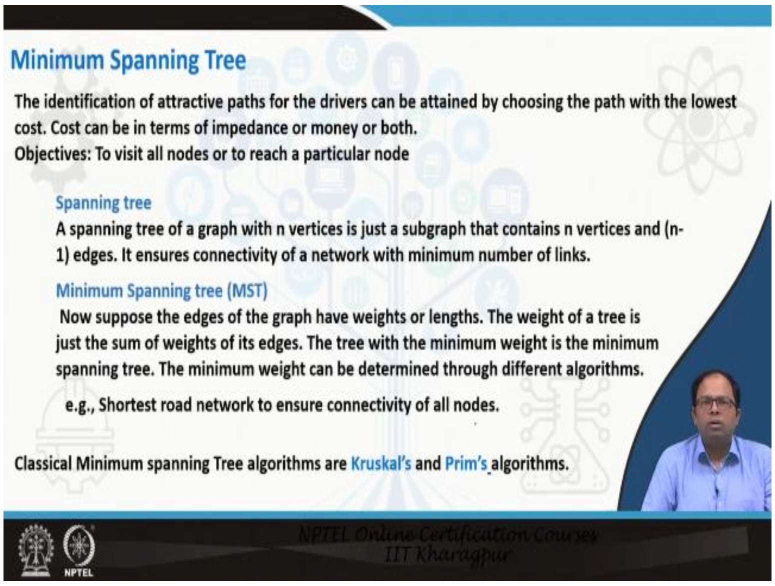
Usually two factors are considered : Time and Monetary cost.

Reasons affecting the route choice decisions of different drivers:

- ❑ Differences in individual perceptions; some may wish to minimize time, others fuel consumption and many a combination of both introducing a variety in route choices.
- ❑ The level of knowledge of alternative routes varies across individuals.
- ❑ Congestion effects affecting shorter routes first and making their generalized costs comparable to initially less attractive routes.

There is a need to identify a set of routes which might be considered attractive to the users prior to traffic assignment. Apart from identification of appropriate routes, the sequence of user addition to these routes must also be taken into consideration. There are several algorithms to determine the attractive routes in terms of cost. ‘Minimum Spanning Tree’ and ‘Shortest Path’ are the two most important algorithms for route determination. Routes are usually identified based on travel time or travel cost or a combination of both. Several factors are thought to influence the choice of route when driving between two points. Following factors like journey time, distance, monetary cost (fuel and others), congestion and queues, type of manoeuvres required, type of road (motorway, trunk road, secondary road), scenery, signposting, road works, reliability of travel time and habit can influence the route choice decisions of the users. But it is difficult to incorporate a generalised cost function based on such exhaustive list of variables. Besides, modelling these variables in traffic assignment model can give rise to complexities. So, only two factors are considered for approximation in route choice, *time* and *monetary cost*. Monetary cost is usually proportional to travel distance. The users can allocate their preferred weights for travel time and distance in most traffic assignment models. The route choice can be estimated based on the generalised cost derived from the weighted sum of the factors mentioned. The route choice decision of different users/drivers can vary due to several reasons. One of the reasons which can influence users'/drivers' decisions is difference in individual perceptions. Some individuals may wish to minimise time whereas others may wish to minimise fuel consumption. Some individuals can look for minimisation of both factors. Difference in perception can lead to different route choice options since the definition of shortest path varies across individuals. The level of knowledge of alternative routes varies across individuals. This will lead to different route options for different individuals based on their level of knowledge. Finally, the congestion effects can also influence route choice decisions. Congestion will affect the shorter routes first making the generalised cost comparable to initially less attractive routes. The routes chosen previously may no longer qualify to be the 'perceived' route choice of the users/drivers. For example, there are two routes. In in one of the routes, the travel time is 10 minutes while the travel time in the other route is 12 minutes. Since the former route has lesser travel time, people will tend to choose the first route. Eventually, it becomes congested and the travel time increases to 30 minutes while the travel time in the other route remains unchanged. Now, the second route becomes more attractive. Therefore, the shortest path can change over time which can lead to change in the route choice.

(Refer Slide Time: 08:38)



Minimum Spanning Tree

The identification of attractive paths for the drivers can be attained by choosing the path with the lowest cost. Cost can be in terms of impedance or money or both.
Objectives: To visit all nodes or to reach a particular node

Spanning tree
A spanning tree of a graph with n vertices is just a subgraph that contains n vertices and (n-1) edges. It ensures connectivity of a network with minimum number of links.

Minimum Spanning tree (MST)
Now suppose the edges of the graph have weights or lengths. The weight of a tree is just the sum of weights of its edges. The tree with the minimum weight is the minimum spanning tree. The minimum weight can be determined through different algorithms.

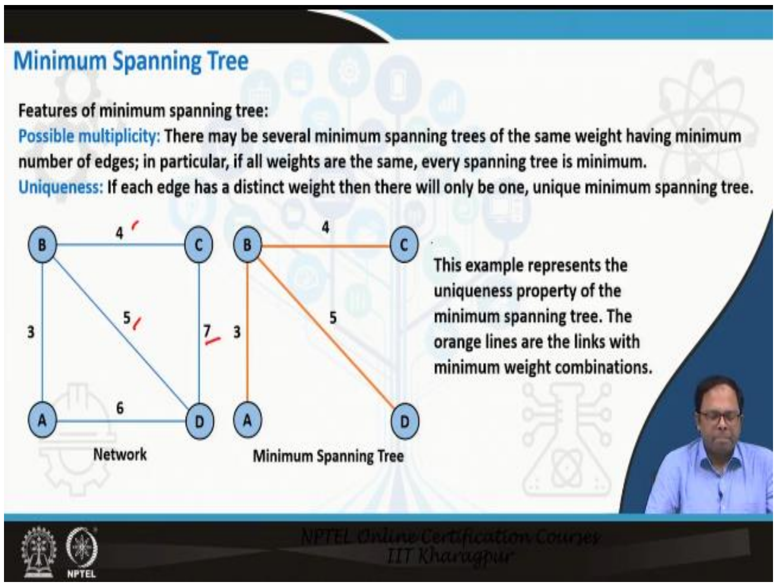
e.g., Shortest road network to ensure connectivity of all nodes.

Classical Minimum spanning Tree algorithms are Kruskal's and Prim's algorithms.

Minimum Spanning Tree

Minimum Spanning Tree (MST) is one of the common algorithms for determining attractive routes in terms of cost. The identification of attractive paths for the drivers can be attained by choosing the path with the lowest cost. Cost can be in terms of impedance or money or both. The objective for MST is to visit all nodes or to reach a particular node. A basic spanning tree (ST) of a graph with *n* vertices is just a subgraph that contains *n* vertices and *n - 1* edges. It ensures connectivity of a network with minimum number of links. We can determine the important links in a network through a ST. We must ensure that ST does not contain *n* edges since it will create a loop. Besides, the edges must be selected such that they do not form a circuit. ST can enable us to visit each particular node by travelling the least number of links. Now for a MST, the cumulative weights of the links of ST must be minimum. Let us assume that, the edges of the graphs have weights or lengths. The weight of a tree is just the sum of the weight of its edges. The tree with the minimum weight is the minimum spanning tree. Several spanning trees connecting all the nodes in a particular graph can exist. But there could be only one spanning tree whose cumulative weight of edges is minimum if weights are unique. MST can be used to determine the shortest road network to ensure connectivity within a network. The minimum weight for the spanning tree can be determined using several algorithms. The classical minimum spanning tree algorithm are Kruskal's algorithm and Prim's algorithm. Some examples have been solved below using these algorithms.

(Refer Slide Time: 11:06)


Minimum Spanning Tree

Features of minimum spanning tree:
Possible multiplicity: There may be several minimum spanning trees of the same weight having minimum number of edges; in particular, if all weights are the same, every spanning tree is minimum.
Uniqueness: If each edge has a distinct weight then there will only be one, unique minimum spanning tree.

This example represents the uniqueness property of the minimum spanning tree. The orange lines are the links with minimum weight combinations.

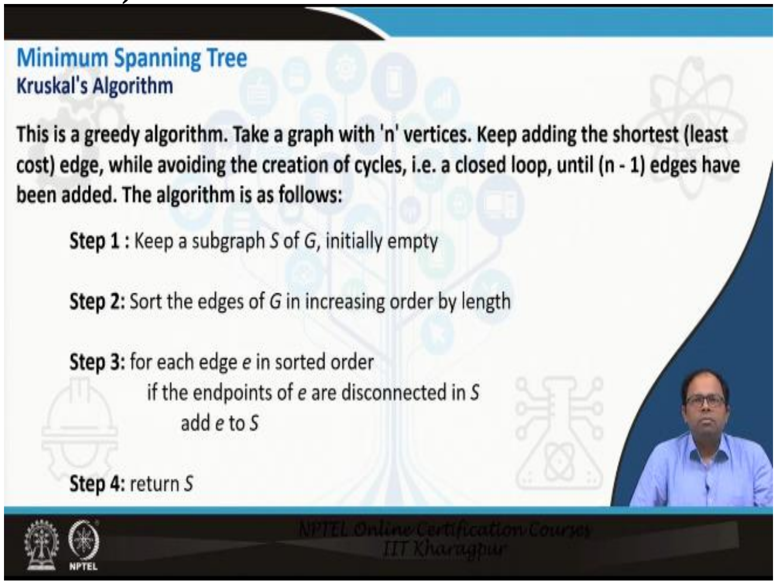
A Minimum Spanning Tree can have two features, *possible multiplicity* and *uniqueness*.

1**. Possible multiplicity-** there may be several minimum spanning trees of the same weight having minimum number of edges. In particular, if all weights are same every spanning tree is minimum. For example, if we give the same weight to all the edges then all the spanning trees will have minimum weight.

2. **Uniqueness-** If each edge has a distinct weight, there will be only one, unique minimum spanning tree, also known as the uniqueness property.

A network with ABCD nodes is given above. Each edge of the network has a distinct weight. These weights are the impedances which are all unique. So, this will generate only one minimum spanning tree as given in the figure above.

(Refer Slide Time: 12:20)


Minimum Spanning Tree
Kruskal's Algorithm

This is a greedy algorithm. Take a graph with 'n' vertices. Keep adding the shortest (least cost) edge, while avoiding the creation of cycles, i.e. a closed loop, until (n - 1) edges have been added. The algorithm is as follows:

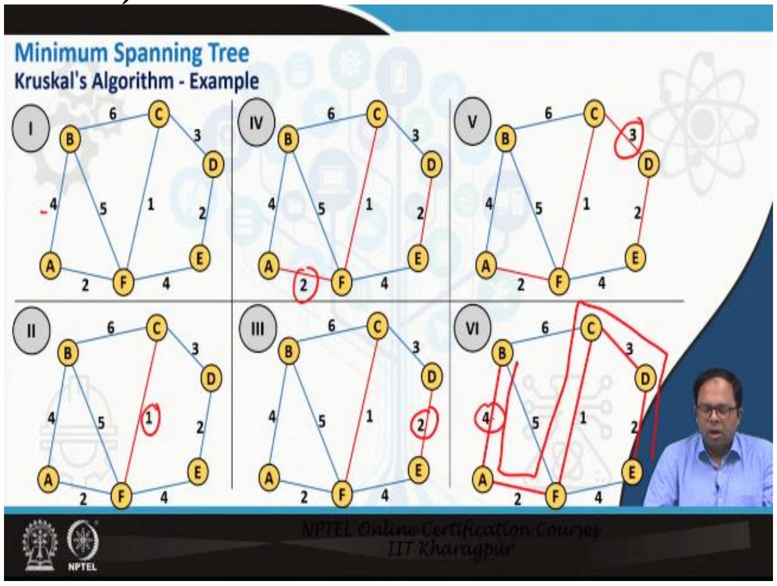
```
Step 1 : Keep a subgraph S of G, initially empty

Step 2: Sort the edges of G in increasing order by length

Step 3: for each edge e in sorted order
            if the endpoints of e are disconnected in S
                add e to S

Step 4: return S
```

Kruskal’s Algorithm

This is one of the classical minimum spanning tree algorithms. This is a greedy algorithm. Let us take a graph with '*n*' vertices. We need to keep adding the shortest (least cost) edge, while avoiding the creation of cycles, i.e. a closed loop, until *(n - 1)* edges have been added. The steps of the algorithm are discussed below.

```
Step 1: Keep a subgraph S of G, initially empty
Step 2: Sort the edges of G in increasing order by length
Step 3: for each edge e in sorted order
            if the endpoints of e are disconnected in S
                add e to S
Step 4: return S
```

(Refer Slide Time: 14:04)



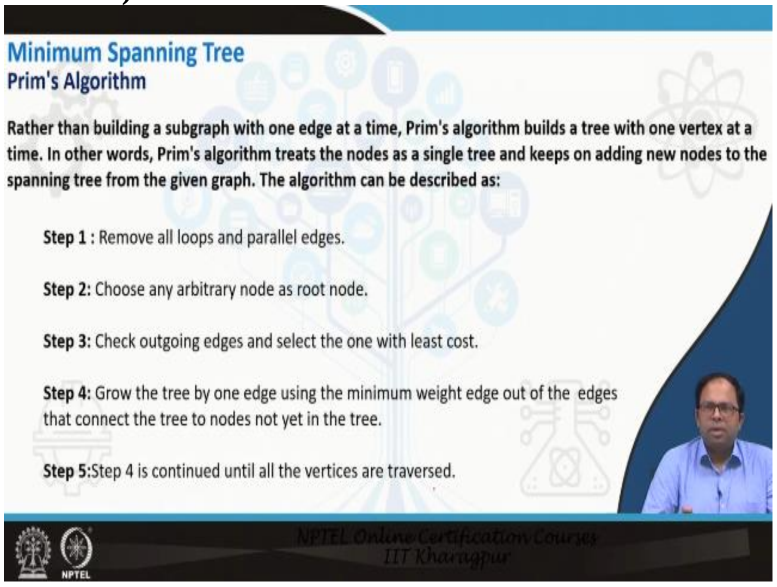
Minimum Spanning Tree

Kruskal's Algorithm - Example

Let us consider a graph with ABCDEF vertices as given above. In the given graph, we have six nodes and eight links. As discussed earlier, a spanning tree will have *n-1* edges. For the given example, the minimum spanning tree will have 5 edges since there are six vertices in the graph. Each edge has a distinct cost or weight. Based on Kruskal’s algorithm, the edges are sorted based on weight. CF is the edge with the minimum weight of 1. So, CF is selected, and the algorithm looks for next minimum weight edge. Now, both DE and AF have the same weight. DE is not connected with edge CF in subgraph S, whereas AF is connected to the edge selected in previous iteration. So, AF is selected and added to the tree. The algorithm continues to add edges until the tree comprises of 5 edges. The iteration stops after the selection of 5th edge because addition of 6th edge will form a closed loop which will deviate the principle of a spanning tree. Finally, the minimum spanning tree from the given graph is

BAFCDE. The total weight of the minimum spanning tree is 4(BA) +2(AF) +1(FC) +3(CD) +2(DE) =12 units.

(Refer Slide Time: 15:46)



Minimum Spanning Tree

Prim's Algorithm

Rather than building a subgraph with one edge at a time, Prim's algorithm builds a tree with one vertex at a time. In other words, Prim's algorithm treats the nodes as a single tree and keeps on adding new nodes to the spanning tree from the given graph. The algorithm can be described as:

Step 1 : Remove all loops and parallel edges.

Step 2: Choose any arbitrary node as root node.

Step 3: Check outgoing edges and select the one with least cost.

Step 4: Grow the tree by one edge using the minimum weight edge out of the edges that connect the tree to nodes not yet in the tree.

Step 5: Step 4 is continued until all the vertices are traversed.

NPTEL Online Certification Courses
IIT Kharagpur

<u>Prim’s Algorithm</u>

This algorithm is another classical minimum spanning tree algorithm. This is a modified form of Kruskal’s algorithm. While we choose edges for Kruskal’s algorithm, nodes are considered for Prim’s algorithm. Rather than building a subgraph with one edge at a time, Prim's algorithm builds a tree with one vertex at a time. In other words, Prim's algorithm treats the nodes as a single tree and keeps on adding new nodes to the spanning tree from the given graph. The algorithm can be described as:

Step 1: Remove all loops and parallel edges. This step is more relevant when some iterations have already been conducted. Also, the creation of loops can also be avoided.

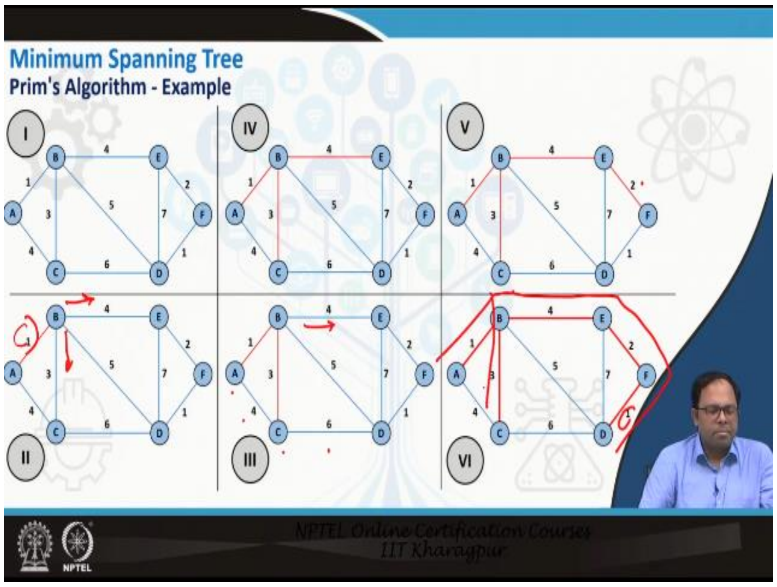
Step 2: Any arbitrary node can be chosen as root node.

Step 3: Check outgoing edges and select the one with least cost.

Step 4: Grow the tree by one edge using the minimum weight edge out of the edges that connect the tree to nodes not yet in the tree. The edges can be connected to the open-ended nodes of the tree so that it could grow further. The tree will grow in the direction of minimum weight edge.

Step 5: Step 4 is continued until all the vertices of the graph are traversed.

(Refer Slide Time: 17:40)



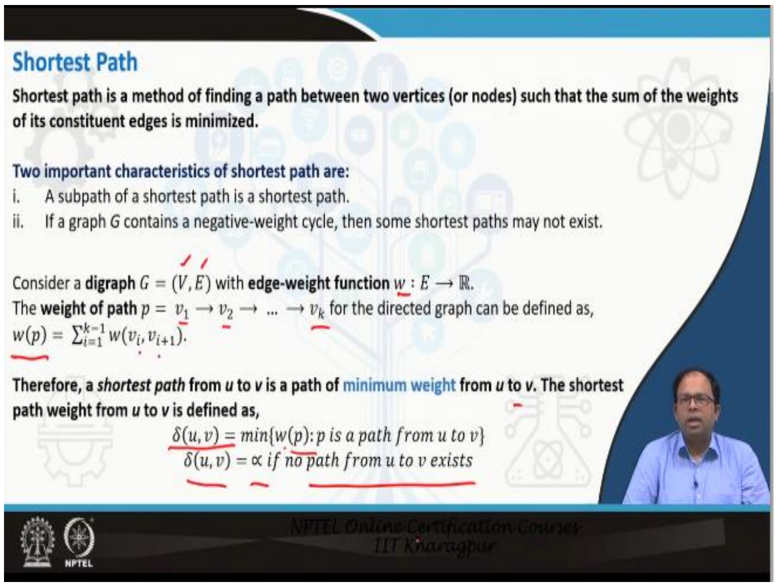
Minimum Spanning Tree

Prim's Algorithm - Example

Let us apply the Prim’s algorithm in the example given above. ABCDEF is the given graph with six nodes. The impedances between the connected nodes are given. We can initiate the algorithm with any one of the nodes. Let us start with A and the outgoing links are checked to identify the least cost edge. There are two outgoing links from A. The edge having the least impedance is AB. So, AB is chosen. Once AB is selected, we need to expand the tree from node B. There are three outgoing links from B. BC is selected since it has the least weight compared to other outgoing links from B. Now if we expand the tree from node C, there will be two outgoing links. If we choose the least cost outgoing link from C, it will create a loop and spanning tree cannot be created. Same holds true for node A. The only direction in which the tree can be expanded is node B. BE being the least cost link is selected in the following step. Node E, F and, D are added to the tree in the subsequent steps. Finally, the minimum spanning tree for the given graph is AB-BC-BE-EF-FD. The total weight of the minimum spanning tree can be estimated as follows:

AB+BC+BE+EF+FD = 1+3+4+2+1 = 11units.

(Refer Slide Time: 19:42)



Shortest Path

Shortest path is a method of finding a path between two vertices (or nodes) such that the sum of the weights of its constituent edges is minimized.

Two important characteristics of shortest path are:

i. A subpath of a shortest path is a shortest path.

ii. If a graph G contains a negative-weight cycle, then some shortest paths may not exist.

Consider a **digraph** $G = (V, E)$ with **edge-weight function** $w : E \rightarrow \mathbb{R}$.
The **weight of path** $p = v_1 \rightarrow v_2 \rightarrow \dots \rightarrow v_k$ for the directed graph can be defined as,
$w(p) = \sum_{i=1}^{k-1} w(v_i, v_{i+1})$.

Therefore, a *shortest path* from *u* to *v* is a path of minimum weight from *u* to *v*. The shortest path weight from *u* to *v* is defined as,

$\delta(u, v) = min\{w(p): p\ is\ a\ path\ from\ u\ to\ v\}$

$\delta(u, v) = \propto if\ no\ path\ from\ u\ to\ v\ exists$

Shortest Path

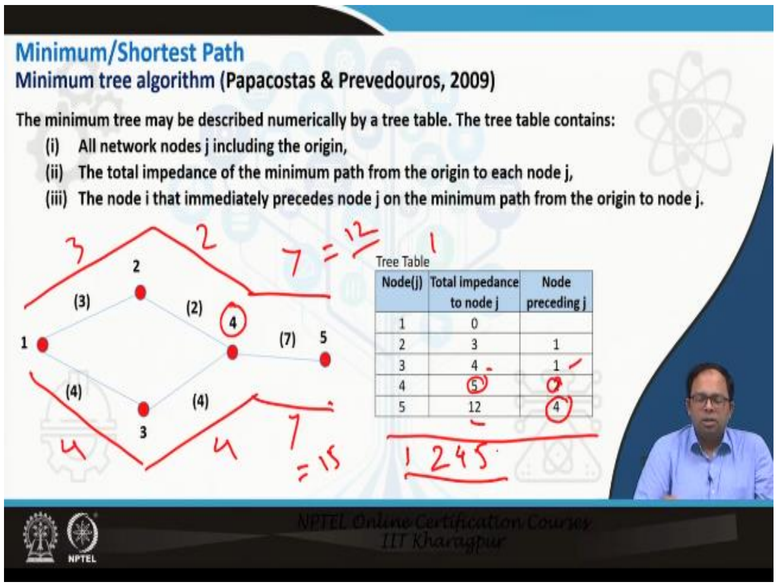
Shortest path is a method of finding a path between two vertices (or nodes) such that the sum of the weights of its constituent edges is minimized. There are two important characteristics of the shortest path. The subpath of a shortest path is also a shortest path because we are already on a shortest path. If a graph *G* contains a negative weight cycle, then some shortest path may not exist. A negative cycle in the graph implies that there is a loop and when the weight of the links in a loop is added, it leads to a negative value of the weight (impedance). Let us consider a digraph where $G = (V, E)$ with edge-weight function w: $E \longrightarrow \mathbb{R}$. So, *V* and *E* are vertices and edges respectively. Each weight function is represented with *w*. The weight of a path $p = v_1 \longrightarrow v_2 \longrightarrow \dots \longrightarrow v_k$ is defined as the summation of weights of all the vertices that is included in the path *p*. The weight function of the path can be represented as,

$$w(p) = \sum_{i=1}^{k-1} w(v_i, v_{i+1})$$

Therefore, a shortest path from *u* to *v* is a path of minimum weight from *u* to *v*. The shortest path weight from *u* to *v* is defined as the minimum of the weight function defined above if a path exists between *u* and *v*. If no path exists between the vertices *u* and *v*, the shortest path is assumed to be infinity because if there is no path then the cost will be maximum. This cost(weight) is represented as infinity. If the shortest path can be identified, the value of infinity can be replaced with a value found for weight (impedance) for that path. The definition for shortest path can be represented as,

$$\delta(u,v)=min\{w(p):p\ is\ a\ path\ from\ u\ to\ v\}$$
$$\delta(u,v)= \propto if\ no\ path\ from\ u\ to\ v\ exists$$

(Refer Slide Time: 22:12)



Minimum/Shortest Path

Minimum tree algorithm (Papacostas & Prevedouros, 2009)

The minimum tree may be described numerically by a tree table. The tree table contains:

(i) All network nodes j including the origin,

(ii) The total impedance of the minimum path from the origin to each node j,

(iii) The node i that immediately precedes node j on the minimum path from the origin to node j.

Tree Table

Node(j)	Total impedance to node j	Node preceding j
1	0	
2	3	1
3	4	1 ✓
4	5	2
5	12	4

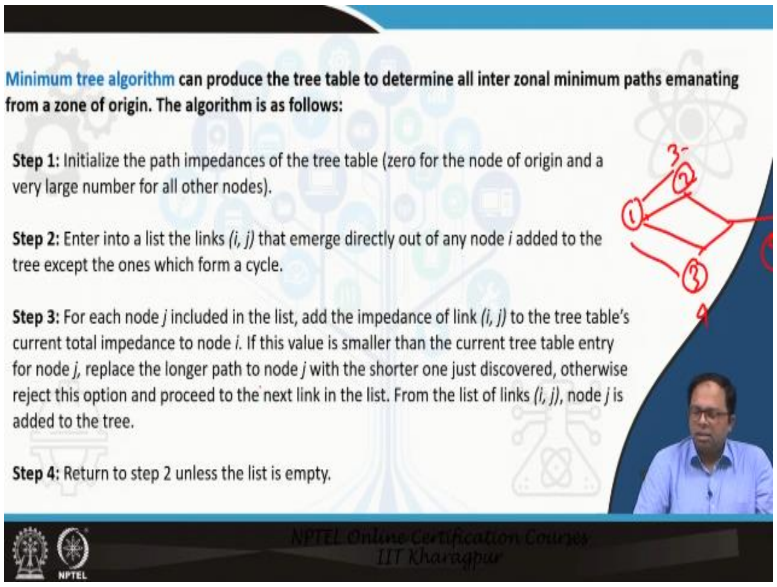
1 2 4 5

NPTEL Online Certification Courses
IIT Kharagpur

The Minimum Tree Algorithm

The minimum tree algorithm can be utilised to determine the shortest path between a given OD pair. Minimum tree algorithm can be described numerically by a tree table. A tree table usually contains three types of information. The tree table must contain all the network nodes *j* including the origin. The total impedance of minimum path from the origin to each node *j* must also be included in the table. Besides, the table must include the information regarding the node *i* that immediately precedes node *j* on the minimum path from origin to node *j*. In the given graph, let us assume the origin is 1. Each node in the given graph is listed for the tree table. The graph consists of five nodes. So, five nodes are included in the column *node(j)* of the tree table. Now the total impedance from origin 1 to each of the nodes listed under this column must be determined. The total impedance must be noted in a separate column in the tree table. If there are multiple values from origin 1 to a node, then the minimum value is noted as the total impedance. In this example, there are two different impedances from origin 1 to node 5, i.e. 12 units and 15 units. Since, 12 is less than 15, the former value is noted as total impedance to node 5. The final column denotes the nodes preceding the nodes to which the total impedances are calculated. Since 12 units has been noted as total impedance, the node preceding node 5 which generates the above value is 4. Therefore, the shortest path for the given graph determined through minimum tree algorithm is 1-2-4-5 with a total weight of 12 units.

(Refer Slide Time: 24:45)



Minimum tree algorithm can produce the tree table to determine all inter zonal minimum paths emanating from a zone of origin. The algorithm is as follows:

Step 1: Initialize the path impedances of the tree table (zero for the node of origin and a very large number for all other nodes).

Step 2: Enter into a list the links *(i, j)* that emerge directly out of any node *i* added to the tree except the ones which form a cycle.

Step 3: For each node *j* included in the list, add the impedance of link *(i, j)* to the tree table's current total impedance to node *i*. If this value is smaller than the current tree table entry for node *j*, replace the longer path to node *j* with the shorter one just discovered, otherwise reject this option and proceed to the next link in the list. From the list of links *(i, j)*, node *j* is added to the tree.

Step 4: Return to step 2 unless the list is empty.

Minimum tree algorithm can produce a tree table to determine all inter zonal minimum paths emanating from zone of origin. The algorithm is given as follows:

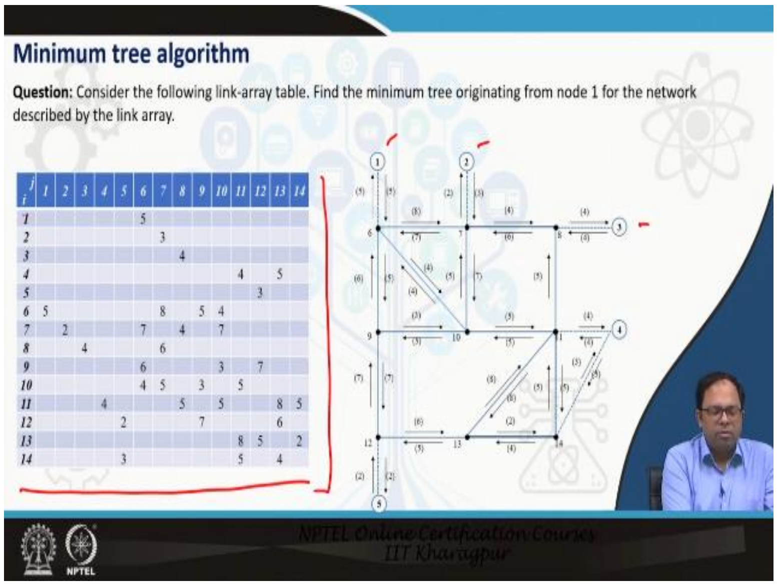
Step 1: Initialize the path impedances of the tree table (zero for the node of origin and a very large number for all other nodes). As discussed earlier, at the initial phase we might not have the knowledge of the existence of a shortest path. So, the node we are considering can be assigned with zero. The remaining nodes can be initiated with the value of infinity.

Step 2: Enter into a list the links *(i, j)* that emerge directly out of any node *i* added to the tree except the ones which form a cycle. For example, for the earlier network, links (1,2) and (1,3) will be added to the list since it emerges from node 1.

Step 3: For each node *j* included in the list, add the impedance of link *(i, j)* to the tree table's current total impedance to node *i*. If this value is smaller than the current tree table entry for node *j*, replace the longer path to node *j* with the shorter one just discovered, otherwise reject this option and proceed to the next link in the list. From the list of links *(i, j)*, node *j* is added to the tree. For example, we initialised the nodes with the value of infinity. Since, the impedance estimated to the nodes are lower than infinity, we can replace the value of infinity with the newly calculated value of impedance. If at any point in iteration, the impedance value calculated to a node is lower than the previously recorded value, the lower value is noted in the column for total impedance. So, the algorithm continues to look for the minimum link and the tree is expanded in the direction of the minimum link.

Step 4: Return to step 2 unless the list is empty. The process is continued until all the nodes in the given network has been taken into consideration.

(Refer Slide Time: 27:42)



Minimum tree algorithm

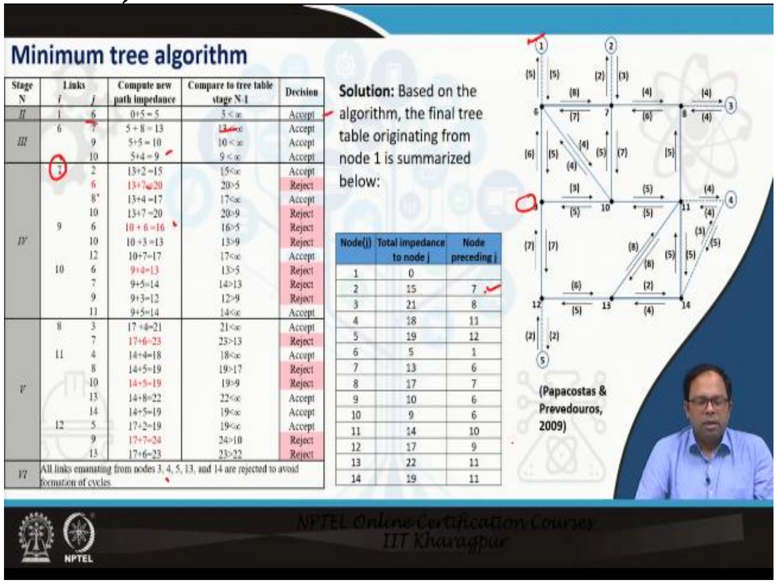
Question: Consider the following link-array table. Find the minimum tree originating from node 1 for the network described by the link array.

i \ j	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1						5								
2							3							
3								4						
4											4		5	
5												3		
6	5						8		5	4				
7		2				7		4		7				
8			4				6							
9						6				3		7		
10						4	5		3		5			
11				4				5		5			8	5
12					2				7				6	
13											8	5		2
14					3						5		4	

NPTEL Online Certification Courses
IIT Kharagpur

Let us see by an example the process to determine the minimum tree for the given network. This is the same network discussed in the previous lecture. The centroid nodes are 1, 2, 3, 4 and 5. The links are either bidirectional or unidirectional. The impedance provided for each link is represented in the scheme matrix. For example, the impedance from node 1 to node 6 is 5. Similarly, from 6 to 1 the impedance is 5, 6 to 7, the impedance is 8 and from 7 to 6 it is 7. Therefore, based on the given network the scheme matrix is prepared. So, if a network is provided the scheme matrix can be derived. Similarly, if scheme matrix is given, the network can be drawn based on the given information in the matrix.

(Refer Slide Time: 28:40)



Minimum tree algorithm

Stage N	Links i	j	Compute new path impedance	Compare to tree table stage N-1	Decision
II	1	6	0+5 = 5	5 < ∞	Accept
III	6	7	5 + 8 = 13	13<∞	Accept
		9	5+5 = 10	10 < ∞	Accept
		10	5+4 = 9	9 < ∞	Accept
IV	7	2	13+2 =15	15<∞	Accept
		6	13+7=20	20>5	Reject
		8	13+4 =17	17<∞	Accept
		10	13+7 =20	20>9	Reject
	9	6	10 + 6 =16	16>5	Reject
		10	10 +3 =13	13>9	Reject
		12	10+7=17	17<∞	Accept
	10	6	9+4=13	13>5	Reject
		7	9+5=14	14>13	Reject
		9	9+3=12	12>9	Reject
		11	9+5=14	14<∞	Accept
V	8	3	17 +4=21	21<∞	Accept
		7	17+6=23	23>13	Reject
	11	4	14+4=18	18<∞	Accept
		8	14+5=19	19>17	Reject
		10	14+5=19	19>9	Reject
		13	14+8=22	22<∞	Accept
		14	14+5=19	19<∞	Accept
	12	5	17+2=19	19<∞	Accept
		9	17+7=24	24>10	Reject
		13	17+6=23	23>22	Reject
VI	All links emanating from nodes 3, 4, 5, 13, and 14 are rejected to avoid formation of cycles				

Solution: Based on the algorithm, the final tree table originating from node 1 is summarized below:

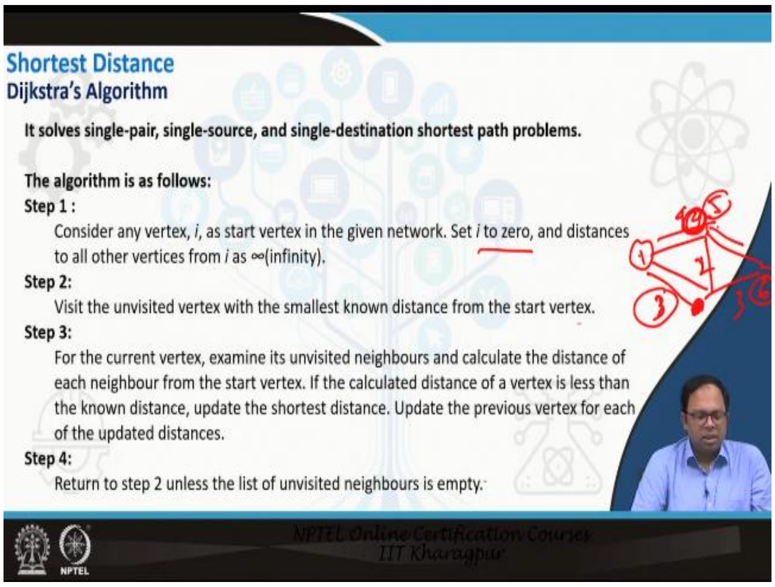
Node(j)	Total impedance to node j	Node preceding j
1	0	
2	15	7
3	21	8
4	18	11
5	19	12
6	5	1
7	13	6
8	17	7
9	10	6
10	9	6
11	14	10
12	17	9
13	22	11
14	19	11

(Papacostas & Prevedouros, 2009)

NPTEL Online Certification Courses
IIT Kharagpur

Once the scheme matrix is prepared from the given network, we can initiate the table building with the node 1. The only link originating from node 1 is 1-6. So, link 1-6 has been listed in the table shown above. The impedance is noted from scheme matrix given earlier and the total impedance is calculated. The given impedance is 5 which is lesser than infinity. So, the impedance value is accepted for link 1-6. Now from node 6 we can expand the tree in three directions either towards 7 or 9 or 10. The same has been listed in the table. Now we need to compute the total impedance till the above node from the starting node, i.e. 1. For 6 to 7, it will be 5 + 8 which is 13 which is less than infinity. Therefore, the new value can be accepted. Similarly, the new impedance to 9 and 10 is 10 and 9 respectively. Now for each of the three nodes, i.e., 7, 9 and 10, we need to expand the tree based on the links originating from these nodes. There are four links originating from node 7. The tree can be expanded to 2 or 6 or 8 or 10. But going back to 6 can create a cycle or a parallel loop. So, this link can be rejected. So, the calculated impedances to 2, 8 and 10 are 15, 17 and 20 respectively. There are three links originating from node 9. These links are towards 6, 10 and 12. The link 9 to 6 can be rejected because it is a parallel link. The updated impedance value to 10 and 12 are 13 and 17 respectively. Based on the impedance value, the connection between 9 and 10 is rejected since 13 is higher than 9 which was calculated for connection between 6 and 10. This process is continued and finally it is found that all the links emanating from nodes 3, 4, 5, 13 and 14 are rejected to avoid formation of cycles. The algorithm terminates since no more nodes can be added to the tree. The final values of impedance from starting node 1 to all other nodes can be listed in a table with information about nodes, total impedance and the preceding nodes. The tree table is created where the total impedance from node 1 to 1 is 0, from 1 to 2 is 15, from 1 to 3 is 21, from 1 to 4 is 18. The preceding node will be identified based on the previous table for which the total impedance value has been found to be minimum. The tree table will comprise of shortest paths to all nodes from the origin node. This is a graph-based method for determining shortest path between any two vertices in a particular network.

(Refer Slide Time: 33:27)



Shortest Distance

Dijkstra's Algorithm

It solves single-pair, single-source, and single-destination shortest path problems.

The algorithm is as follows:

Step 1 :
Consider any vertex, *i*, as start vertex in the given network. Set *i* to zero, and distances to all other vertices from *i* as ∞(infinity).

Step 2:
Visit the unvisited vertex with the smallest known distance from the start vertex.

Step 3:
For the current vertex, examine its unvisited neighbours and calculate the distance of each neighbour from the start vertex. If the calculated distance of a vertex is less than the known distance, update the shortest distance. Update the previous vertex for each of the updated distances.

Step 4:
Return to step 2 unless the list of unvisited neighbours is empty.

NPTEL Online Certification Courses
IIT Kharagpur

Dijkstra's Algorithm

This algorithm solves a single pair single source and a single destination shortest path problem. It will determine the shortest path between a single OD pair. The drawback of this algorithm is that it does not determine the shortest path from an origin to all the destinations. The algorithm can be described as follows:

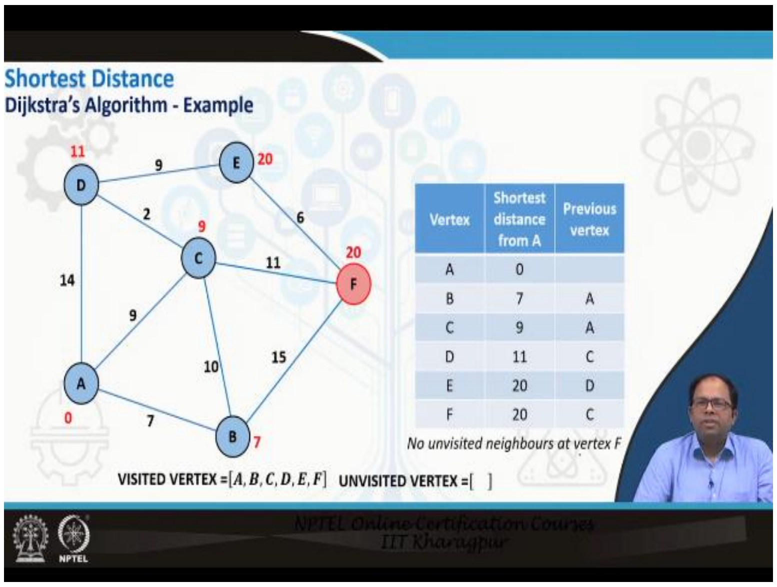
Step 1: Consider any vertex, *i*, as start vertex in the given network. Set *i* to zero, and distances to all other vertices from *i* as ∞(infinity) similar to the minimum tree algorithm.

Step 2: Visit the unvisited vertex with the smallest known distance from the start vertex. For example, if the impedance for two links originating from a vertex is 3 and 4 units, then the link with impedance of 3 units will be selected since it is the smallest known distance (impedance).

Step 3: For the current vertex, examine its unvisited neighbours and calculate the distance of each neighbour from the start vertex. If the calculated distance of a vertex is less than the known distance, update the shortest distance. Update the previous vertex for each of the updated distances. For example, the current vertex is 2 and its neighbours are 3 and 4. The distance of 3 and 4 from starting vertex 1 through vertex 2 is 5 and 3 respectively, while the distance of vertex 3 through vertex 5 is 7. Since 5 is less than 7, vertex 3 is updated with the shortest distance i.e. 5. Therefore, we also need to take care of the vertex that was recorded in the previous iteration.

Step 4: Return to step 2 unless the list of unvisited neighbours is empty.

(Refer Slide Time: 36:03)



Shortest Distance

Dijkstra's Algorithm - Example

Vertex	Shortest distance from A	Previous vertex
A	0	
B	7	A
C	9	A
D	11	C
E	20	D
F	20	C

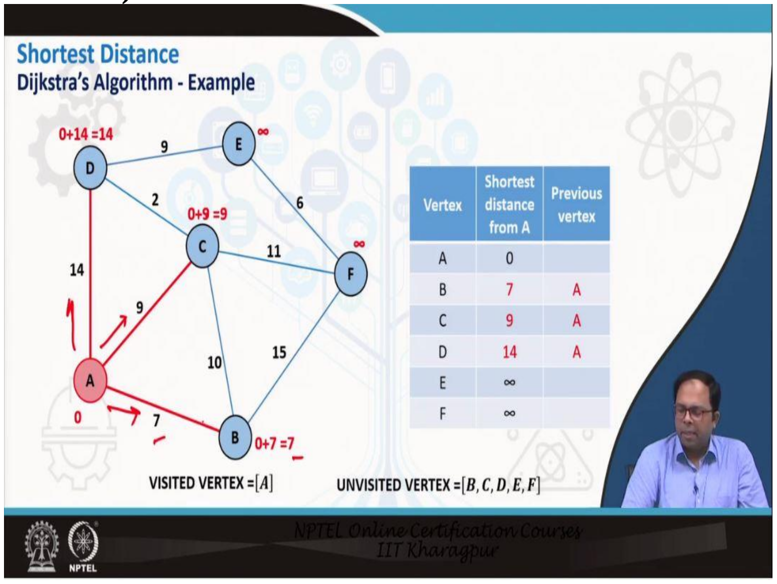
No unvisited neighbours at vertex F

VISITED VERTEX =[A, B, C, D, E, F] UNVISITED VERTEX =[ ]

NPTEL Online Certification Courses IIT Kharagpur

Let us solve the algorithm through an example. In this graph, ABCDEF these are six vertices. Based on the algorithm, let us initiate with vertex A. So, the value of A is set as 0 and all other vertices are set to infinity. A table is also created to list all the vertices and to note the shortest distance from the origin vertex A. Besides, the preceding vertex of a vertex is also noted. For the initial phase, the shortest distance of all vertices from vertex A will be ∞(infinity). No previous vertices can be recorded in the first step.

(Refer Slide Time: 36:44)



Shortest Distance

Dijkstra's Algorithm - Example

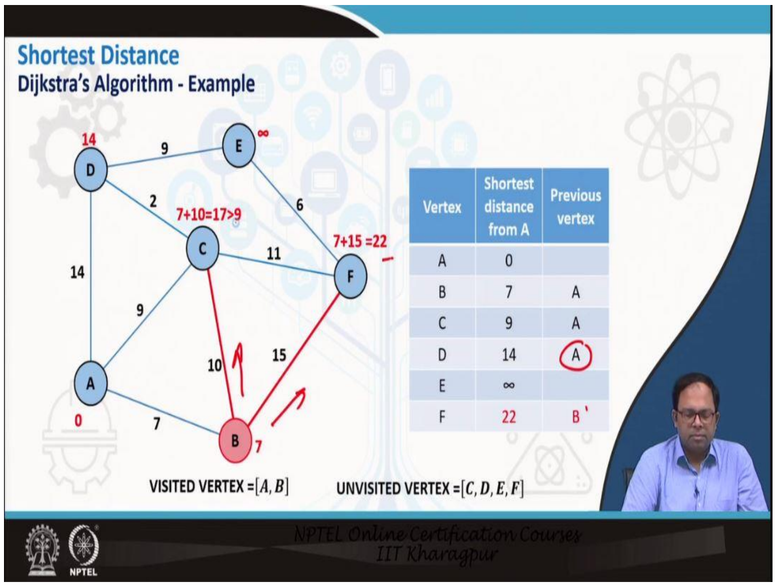
Vertex	Shortest distance from A	Previous vertex
A	0	
B	7	A
C	9	A
D	14	A
E	∞	
F	∞	

VISITED VERTEX =[A] UNVISITED VERTEX =[B, C, D, E, F]

NPTEL Online Certification Courses IIT Kharagpur

One origin vertex A is fixed, its neighbours can be identified. In this example, neighbours of A are vertex B, C and D. The weight of the vertex B, C and D is recorded as (0+7), (0+9) and (0+14) respectively. The same is recorded in the table. The previous vertex of all these vertices is recorded as A. The vertex E and F continues to have the same weight, i.e., infinity since these vertices are not the neighbours of A.

(Refer Slide Time: 37:28)



Shortest Distance
Dijkstra's Algorithm - Example

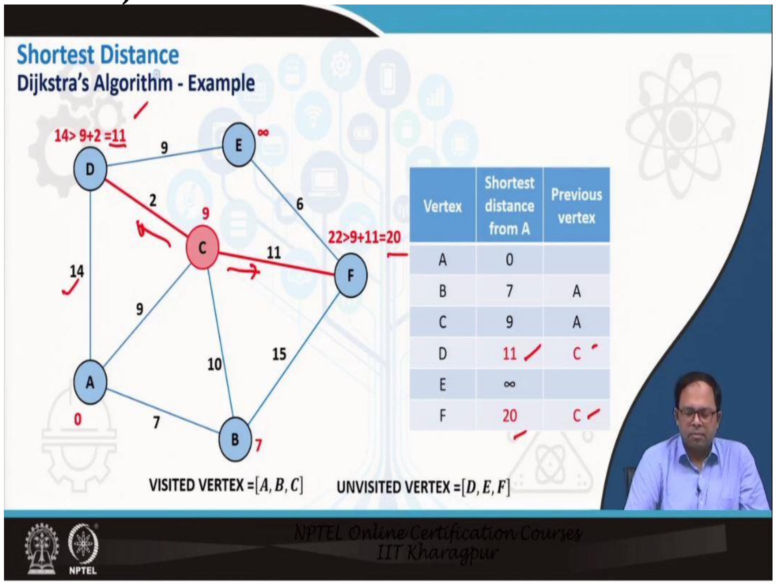
Vertex	Shortest distance from A	Previous vertex
A	0	
B	7	A
C	9	A
D	14	A
E	∞	
F	22	B

VISITED VERTEX =[A, B] UNVISITED VERTEX =[C, D, E, F]

NPTEL Online Certification Courses
IIT Kharagpur

The vertex with the smallest known weight is B. So, vertex B is selected as current vertex. Now, the neighbours of B are A, C and F. Since, the vertex A is already visited, A cannot be included in the neighbour list. The new impedance of vertex C and F through B is 17 and 22 respectively. Now the new impedance for C is greater than the previous impedance. So, the previous impedance for vertex C will be maintained. The new impedance for vertex F is 22 which is less than infinity. So, the new impedance is maintained for vertex F. D and E are not the neighbouring vertices of B. So, the previous records are maintained. In this step, information related to vertex F is only updated in the table.

(Refer Slide Time: 38:46)



Shortest Distance
Dijkstra's Algorithm - Example

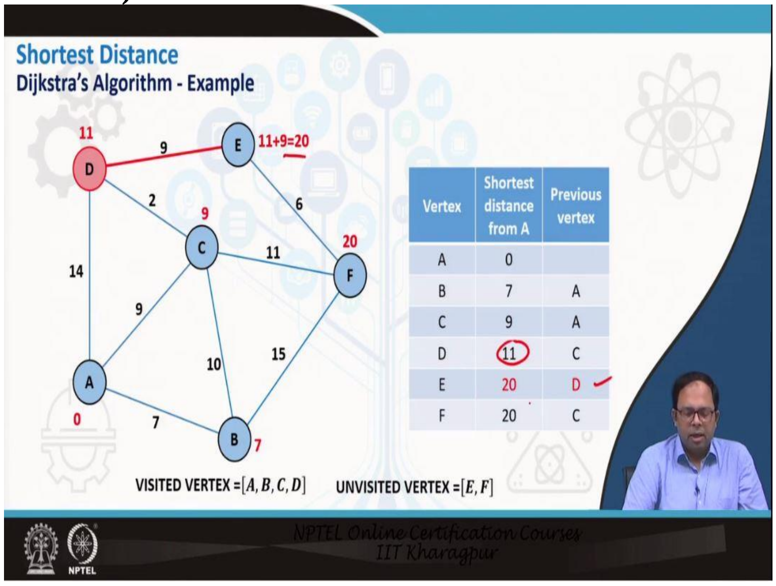
Vertex	Shortest distance from A	Previous vertex
A	0	
B	7	A
C	9	A
D	11	C
E	∞	
F	20	C

VISITED VERTEX =[A, B, C] UNVISITED VERTEX =[D, E, F]

NPTEL Online Certification Courses
IIT Kharagpur

Now, from vertex B the closest vertex must be selected as the current vertex. Out of the two neighbouring vertices, C and F, C is the closest vertex. So, C is selected as the current vertex. The neighbours of C are A, B, D and F. While D and F are unvisited neighbours, A and B are the visited neighbours. So, only D and F are considered to determine the shortest path. The new distance of vertex D and F from A is (9+2) /11 and (9+11)/20 respectively. Both the updated distances are lower than the distances estimated in previous iterations. So, the vertex D and F are updated with the new distance. Since the distance is updated, the preceding vertex of both the vertices is changed. These changes are recorded in the table as shown above.

(Refer Slide Time: 39:41)



Shortest Distance

Dijkstra's Algorithm - Example

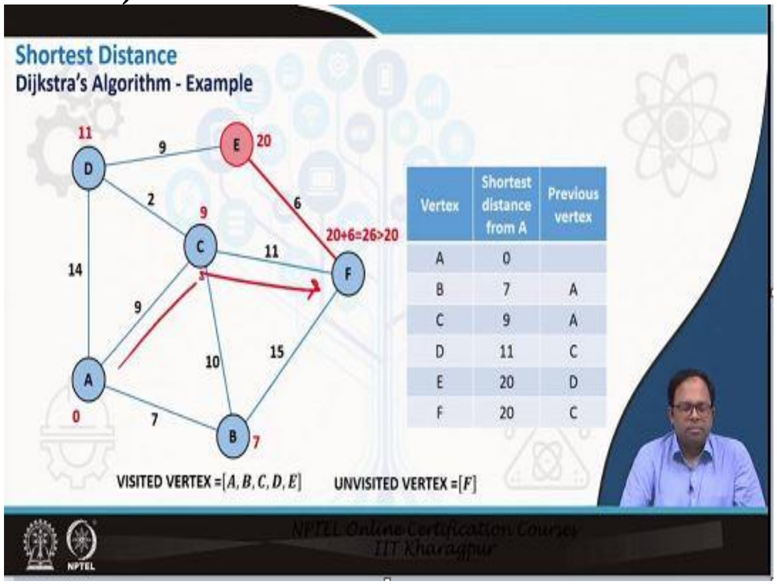
Vertex	Shortest distance from A	Previous vertex
A	0	
B	7	A
C	9	A
D	11	C
E	20	D
F	20	C

VISITED VERTEX =[A, B, C, D]   UNVISITED VERTEX =[E, F]

NPTEL Online Certification Courses
IIT Kharagpur

Now, D being the closest vertex from C, it is selected in the next step. The only unvisited neighbour of D is E. The new distance for E is 20 which is less than the previous value, i.e. infinity. So, the shortest distance of E from A through D is updated and the previous vertex for this path is also recorded. All other values in the table remains unaltered.

(Refer Slide Time: 40:08)



Shortest Distance

Dijkstra's Algorithm - Example

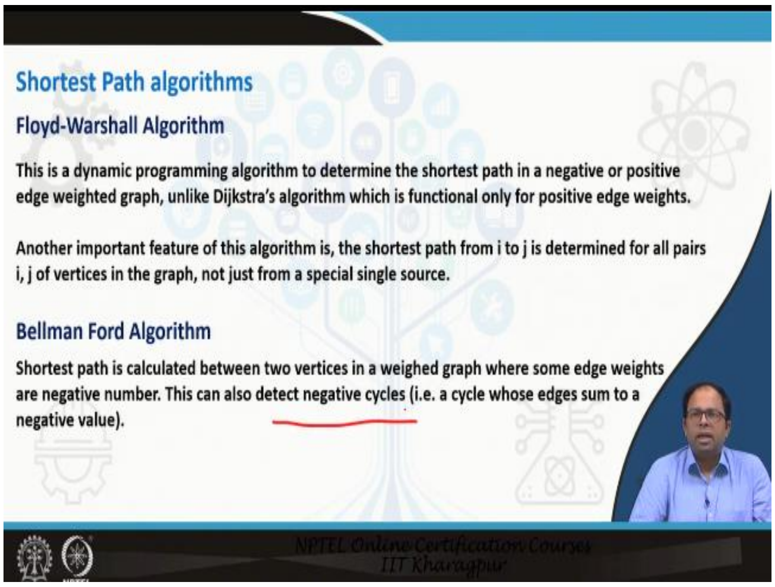
Vertex	Shortest distance from A	Previous vertex
A	0	
B	7	A
C	9	A
D	11	C
E	20	D
F	20	C

VISITED VERTEX =[A, B, C, D, E]   UNVISITED VERTEX =[F]

NPTEL Online Certification Courses
IIT Kharagpur

In the following step, E is selected. The only unvisited neighbour of E is F. The new shortest distance of F from A through E is (20+6) i.e. 26. The new value is higher than the previous value for F. So, the shortest path recorded for F from A remains unchanged. So, now all the vertices have been visited and the list for unvisited vertex is empty. The algorithm can be terminated at this stage. So, the shortest path has been determined from A to all other vertices. For example, the shortest path from A to D is, A-C-D. The shortest path from A to F is A-C-F. At F, there are no more shortest paths since there are no more new vertices.

(Refer Slide Time: 41:11)



Shortest Path algorithms

Floyd-Warshall Algorithm

This is a dynamic programming algorithm to determine the shortest path in a negative or positive edge weighted graph, unlike Dijkstra's algorithm which is functional only for positive edge weights.

Another important feature of this algorithm is, the shortest path from i to j is determined for all pairs i, j of vertices in the graph, not just from a special single source.

Bellman Ford Algorithm

Shortest path is calculated between two vertices in a weighed graph where some edge weights are negative number. This can also detect negative cycles (i.e. a cycle whose edges sum to a negative value).

NPTEL Online Certification Courses
IIT Kharagpur

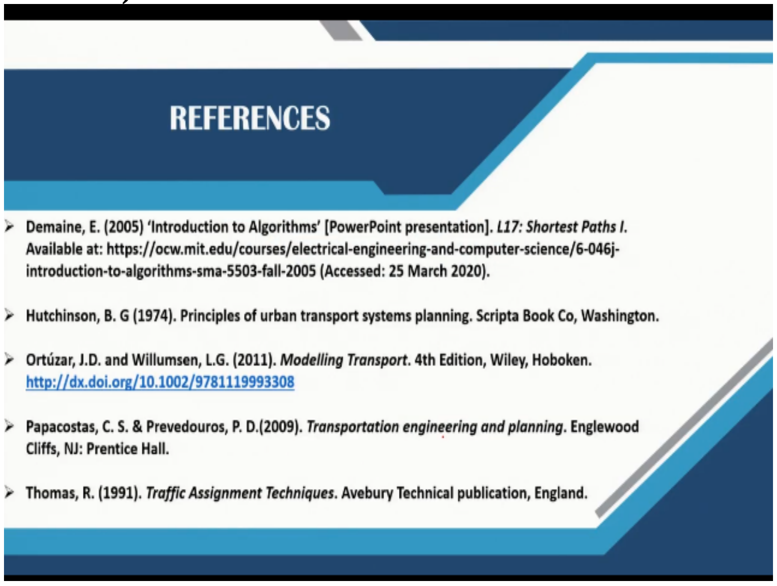
<u>Other Shortest Path Algorithm</u>

In addition to Dijkstras algorithm, there are other algorithms which can also determine the shortest paths for a network. These algorithms can consider both positive and negative weighted graphs unlike Dijkstras algorithm which only takes positive weighted graphs into consideration. These algorithms are Floyd-Warshall algorithm and Bellman Ford algorithm.

Floyd-Warshall algorithm - This is a dynamic programming algorithm to determine the shortest path in a negative or positive edge weighted graph, unlike Dijkstra's algorithm which is functional only for positive edge weights. Usually, Dijkstra's algorithm is sufficient for transportation network since negative edge weighted graphs are unusual in transportation. Another important feature of this algorithm is, unlike dijkstra's algorithm,the shortest path from *i* to *j* is determined for all pairs *i,j* of vertices in the graph and not just from a special single source.

Bellman Ford algorithm - Bellman Ford algorithm is a slower in terms of computation times compared to the Floyd Warshall algorithm. Shortest path is calculated between two vertices in a weighed graph where some edge weights are negative number. The advantage of this algorithm over the previous algorithm is it can also detect negative cycles (i.e. a cycle whose edges sum to a negative value).

(Refer Slide Time: 43:15)

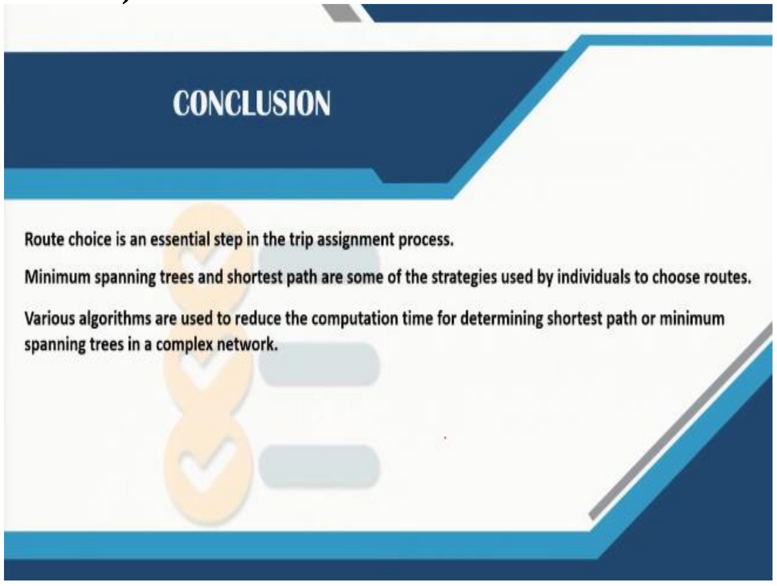


REFERENCES

- Demaine, E. (2005) 'Introduction to Algorithms' [PowerPoint presentation]. *L17: Shortest Paths I*. Available at: https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-046j-introduction-to-algorithms-sma-5503-fall-2005 (Accessed: 25 March 2020).
- Hutchinson, B. G (1974). Principles of urban transport systems planning. Scripta Book Co, Washington.
- Ortúzar, J.D. and Willumsen, L.G. (2011). *Modelling Transport*. 4th Edition, Wiley, Hoboken. http://dx.doi.org/10.1002/9781119993308
- Papacostas, C. S. & Prevedouros, P. D.(2009). *Transportation engineering and planning*. Englewood Cliffs, NJ: Prentice Hall.
- Thomas, R. (1991). *Traffic Assignment Techniques*. Avebury Technical publication, England.

So, these are some of the references you can use.

(Refer Slide Time: 43:21)



CONCLUSION

Route choice is an essential step in the trip assignment process.

Minimum spanning trees and shortest path are some of the strategies used by individuals to choose routes.

Various algorithms are used to reduce the computation time for determining shortest path or minimum spanning trees in a complex network.

<u>Conclusion</u>

Route choice is an essential step in the trip assignment process. Trip assignment cannot be conducted without route choice determination. The minimum spanning tree and shortest path are some of the strategies used by individuals to choose the attractive routes. There are various algorithms that are used to reduce the computation time for determining shortest path or minimum spanning trees in a complex network. This lecture has covered some of these algorithms. Other algorithms can be explored as well.

Thank you!