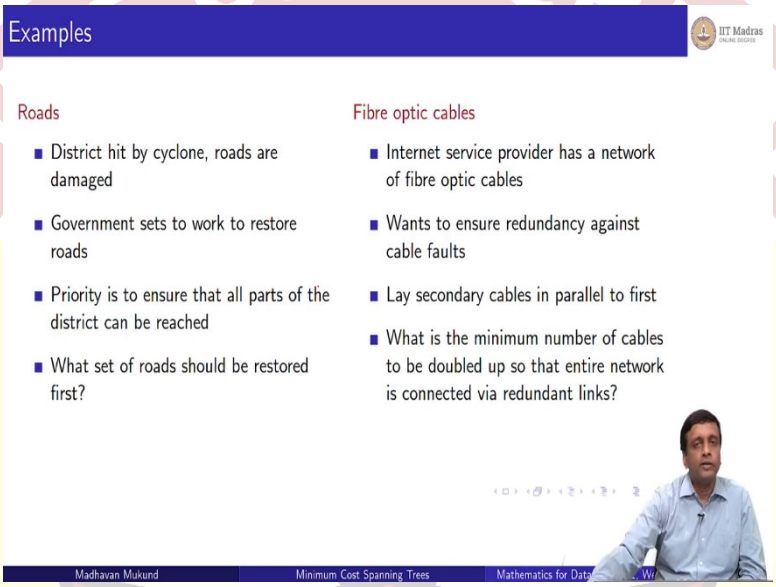
IIT Madras

ONLINE DEGREE

Mathematics for Data Sciences 1
Professor. Madhavan Mukund
Chennai Mathematical Institute
Lecture No. 12.5
Minimum Cost Spanning Tress

So, we have looked at shortest paths, both the single source version and the all-pairs version with and without negative weights. And now in the context of weighted graphs, we move to a different problem, which is the problem of computing minimum cost spanning trees.

(Refer Slide Time: 00:31)



So, to motivate this problem, let us look at a couple of examples. So, here is the first example. So, supposing you are in a district, which has been hit by a cyclone, and many of the roads are damaged. So, immediately after the cyclone, of course, the first priority is to restore the roads. But you also want to restore the roads in such a way that everybody can move around as quickly as possible.

So, you do not want to start at one end of the district and move sequentially to the other end of the district, what you want to do is prioritize the roads to be repaired, so that everybody is connected to everybody as fast as possible. So, which set of roads should you restore, so that connectivity across the district is maximally restored, rather than individual parts being connected and other parts being disconnected?

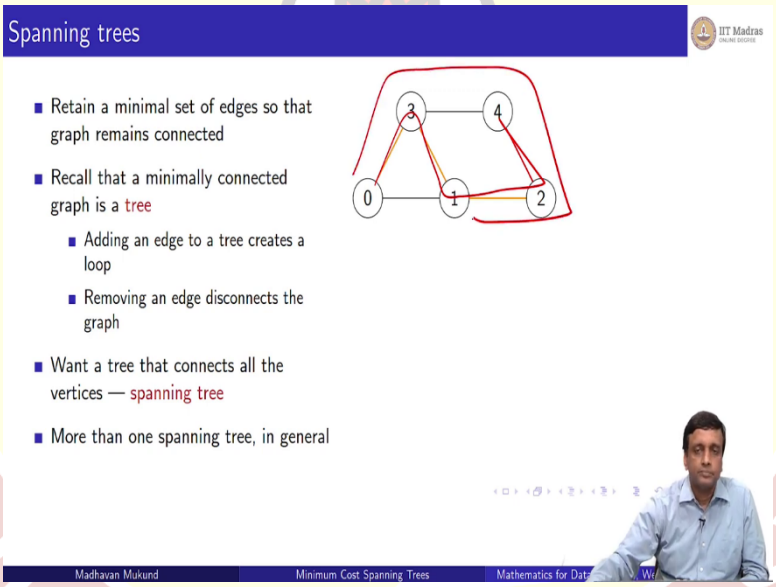
Here is another context. So, suppose you are an internet service provider. So, you provide internet connectivity to a large number of customers in different cities, and then your customers are demanding reliability. They are saying that in some cases, because of some damage, either

due to an accident or due to some construction or something, if a cable between two cities gets cut, then their services cut.

So, you want to lay a parallel cable to ensure that if one cable is cut, the other cable still works. But at the same time, you want to do this in such a way that you do not spend too much laying parallel cables everywhere, you do not want to double up every cable in your network, you want to double up sufficient number of cables, such that between any two locations on your network, there is a redundant route.

So, you are not obliged to put a double cable between every pair of nodes or every pair of cities on a network, only enough of them so that everyone is guaranteed to be connected to everyone else even if one link fails. So, this is a related problem. So, these are both problems will feed into this problem of finding a spanning tree.

(Refer Slide Time: 02:15)



So, a spanning tree essentially asks us how do we take a graph which is connected and retain a minimum set of edges so that it remains connected. So, a minimum set of edges that is connected is a tree. So, we said that a tree is a connected acyclic graph, and we will talk about trees in more detail in this lecture. But the intuition is that if you want to connect n nodes in a minimal way, what you end up doing is connecting them in such a way that there are no redundant paths, there are no cycles, so this is a tree.

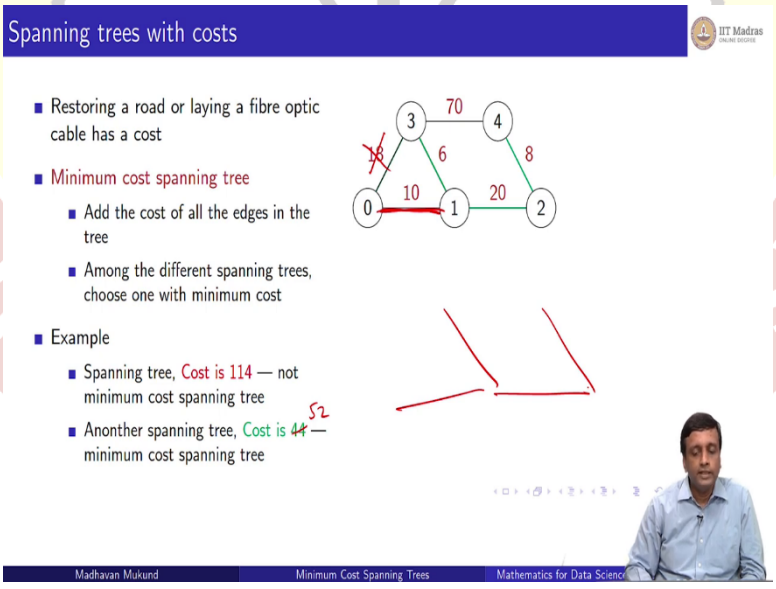
So, if you add an edge to a tree, you add redundancy, so you get a loop. If you remove an edge from a tree because it is kind of minimal, if you remove an edge from a tree, the tree will fall apart, it is no longer going to be connected that is why it is a minimal acyclic connected graph.

So, what we want in this situation both in the road situation and in the telecom situation, that ISP situation is that we want to connect a subset of the nodes. Now, we want to say, we want to restore a subset of the roads, or we want to double up a subset of the links such that everything is connected to everything.

So, we want to find a subset of the edges in the original graph, which if I deal with them, either by repairing the roads or by upgrading them to a double link, I will end up connecting everything to everything. So, here on the right for instance is a graph and this red thing is a spanning tree. So, it is a spanning tree. So, spanning tree is something that connects all the vertices, so it spans the graph, it touches every vertex in the graph and it is a tree so it is a subset of the edges it touches every vertex in the graph, it is a tree so, the red edges here form a spanning tree.

Now, this spanning tree is not going to be unique. So, here is another spanning tree. So, this orange where it is now also this one is also a spanning tree. So, the earlier one was one which went this way and now we have one which goes this way. So, we have two different spanning trees you can have multiple spanning trees. So, you could also have a spanning tree which goes like this for it this is also spanning tree.

(Refer Slide Time: 04:22)



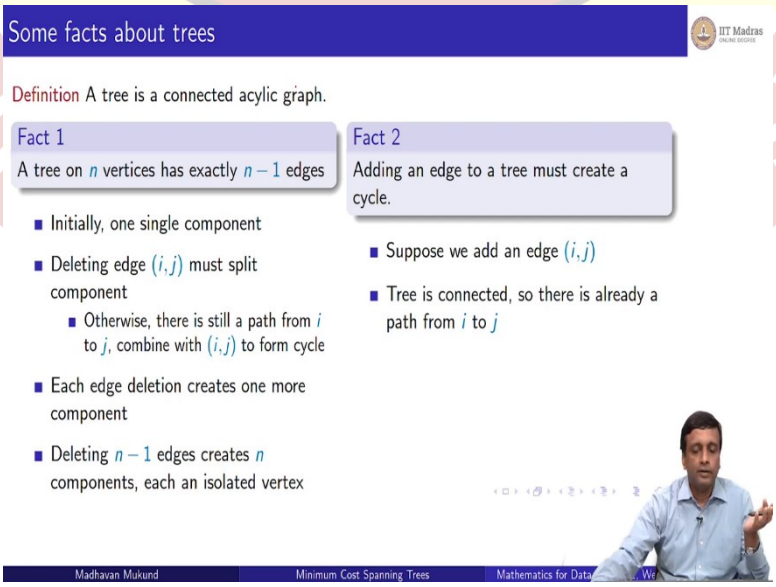
So, our interest is weighted graphs. So, supposing our goal is not just that we want to find a subset of roads to fix or a subset of edges telecom links to double up, but there is a cost associated with this. So, laying a road depending on the location and various other features, laying road may not be the same cost all over the place. Similarly, there may be difficulty in laying cables in some places, not in other places.

So, now if we have a difficulty or a cost or some kind of measure associated with every edge that we want to deal with, can I find a shortest or minimum cost way of doing this? So, I will want to find a minimum cost spanning tree? So, it is not. So, we saw that there could be many different spanning trees. So, it is not just any old spanning tree, but a spanning tree, who if I look at the cost of the all the edges, which I am adding to that tree, so that is the way I am going to define.

So, remember, when we had a shortest path and a weighted graph, we added the cost of all the edges in the path. So, here we are constructing a tree in a graph and we are going to take all the edges that fall into the tree and say that is the total cost I am going to spend if I am going to build this tree, if I am going to repair these roads, or if I am going to develop these cables, this is going to be my total cost. So, I want to find the minimum spanning tree and this is called a minimum cost spanning tree.

So, if I look at this example, for instance, so here is one spanning tree. So, this spanning tree has cost 18+ 6 24+ 17 94+ 20 14. Now, we can easily check that this is not a minimum cost spanning tree, in this case, is small graph, because we can construct this green tree for instance, which has a shorter cost. So, this is 18+ 6 24, 44, this is not this is 52. So, this is actually 28+ 6 44 52. But if I take out this and I put this instead, so this is also a spanning tree. So, this is a spanning tree also for this graph and this you will check has cost 44. It is 28+ 16 is 44. So, among all the trees that I can draw on this particular graph, it turns out that 44 is the best one.

(Refer Slide Time: 06:28)



So, in order to come up with algorithms or strategies to discover minimum costs spanning trees, we will do some basic facts about trees and these will be useful in general. So, it is very good

to write them down once and for all so that you know them and you remember them so that you are aware of what you are doing when you are dealing with trees. So, as far as we are concerned, the basic definition of a tree is that it is a connected graph and it is acyclic this is all we are told you are given n vertices, the graph on n vertices is connected, and it has no cycle. So, we are assuming this is an undirected graph. So, it has no undirected cycles. What can you conclude from this?

So, this is a tree, a tree is just a connected graph, which is acyclic. So, the first thing we will conclude is that if the graph had n vertices, then the tree must have exactly n-1 edges, not more, not less, it has exactly n-1 edges. So, here is one argument why that is the case. So, we know that this graph is connected. So, remember that we talked about connected components. So, as an undirected graph, this whole graph, that I am given initially, is a single component because it is connected.

But now I also know that it is acyclic. So, it is acyclic, I claim that if I delete an edge, then it must disconnect the graph because if it did not disconnect the graph, then if I delete an edge, and I can still go across that edge from i to j via some other route, so I have deleted an edge ij, in the tree that is given to me, before I deleted the edges connected after deleted if it is still connected, it means there is still a way to go from it.

But if there is still a way from a go from i to j, does not involve this edge ij. So, if I add back this edge, ij, then I can go from i go to j by the other path and then come back on the site. So, there is like, but I also know that the tree is acyclic. So, therefore, it must be the case that when I remove an edge from a tree, the tree will fall apart into two components. It cannot fall by more than two components, because there is only one edge only connects two parts. So, the whole thing was one component, it is like a cut one thread, and the whole thing falls apart and two pieces.

Now I have two connected things. I cut one more edge, what will happen, one of these two will fall into two more things. So, every time I cut an edge, I create an extra component, though I make one component or two components, the other components are unchanged. So, every time I delete an edge, I am going to create one more component. But how many components can I create?

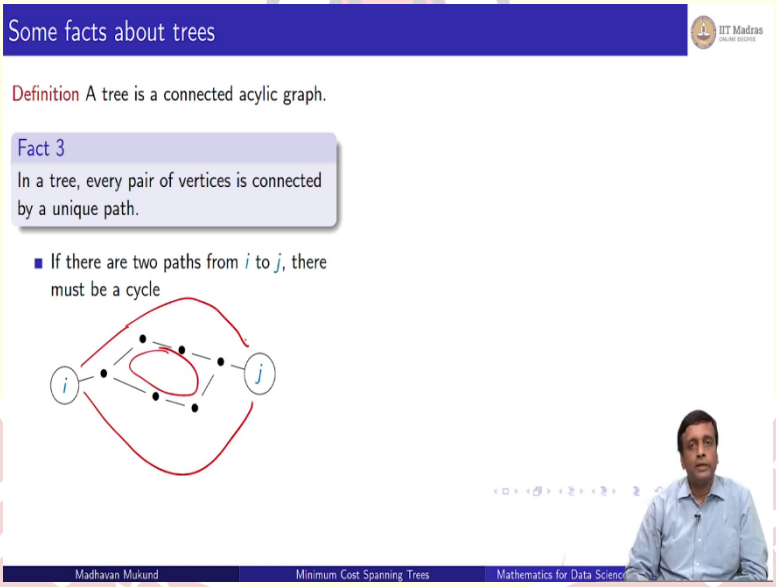
Well, I claim that at most, I can create n components because there are only n vertices. Finally, the minimum component is disconnected vertices isolated by itself with no connections. So, I started with one component, and I ended up with n components. And every time I did+ 1, so

how many times can I go from 1 to n to n+ 1, n-1 times, so I could only delete n-1 edges. So, this is one argument saying that every tree on n vertices must have exactly n-1 edges.

So, this says there are no more than n-1. And you can obviously argue that if I had fewer than n-1, then at some point, this thing would have got disconnected earlier. And that is also a contradiction. The other flip side to this is that if I add something to this tree, then it will create a cycle. So, in some sense, this is a minimum connected graph, it is a minimal connected graph, that is adding more edges will only complicate the situation in terms of connectivity.

So, adding an edge is essentially symmetric to what we said before. So, we said before that if you delete an edge, you must split the graph into two otherwise it would have been a cycle. Now if I add an edge, I know that i and j are already connected in the tree. So, if I add an edge by the same logic, I have created a cycle. So, therefore, whenever I add an edge to a tree, it creates a cycle.

(Refer Slide Time: 10:13)



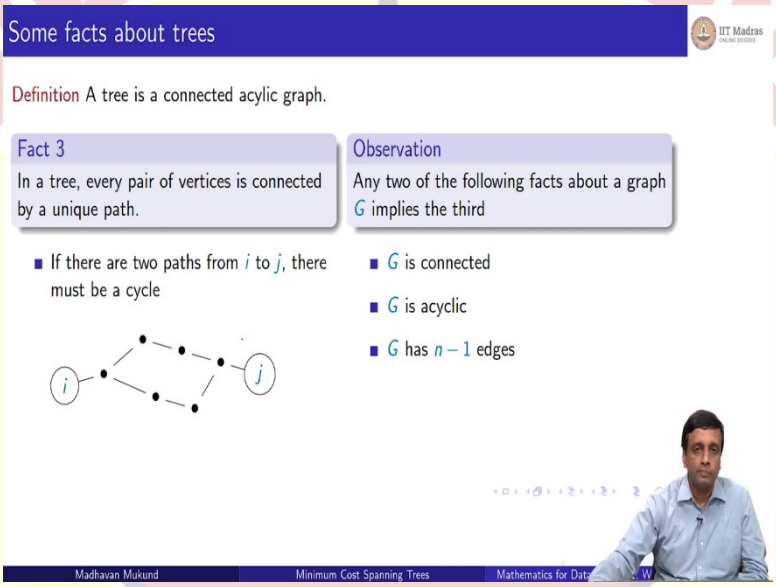
The third fact is that between any two points in a tree, there is only one way to go. There is only one path between any two vertices in a tree. This is not true, in general, as we have seen in many graphs, you can go many ways. For instance, when you are calculating shortest paths, we found alternative paths, which got us shorter weights, and so on. But in a tree, this is not possible in a tree, I can only go from i to j in one way it is connected, guaranteed, but it is connected by only one way.

So, we will just look at a pictorial thing. So, supposing there are two ways to go from i to j. So, the argument is that if there are two ways to go from i to j, then somewhere in between, there

must be something like this, a structure like this, where the two paths diverged, and the two, so the two paths might divergent at i and j itself, it might be that I have to completely separate paths i to j.

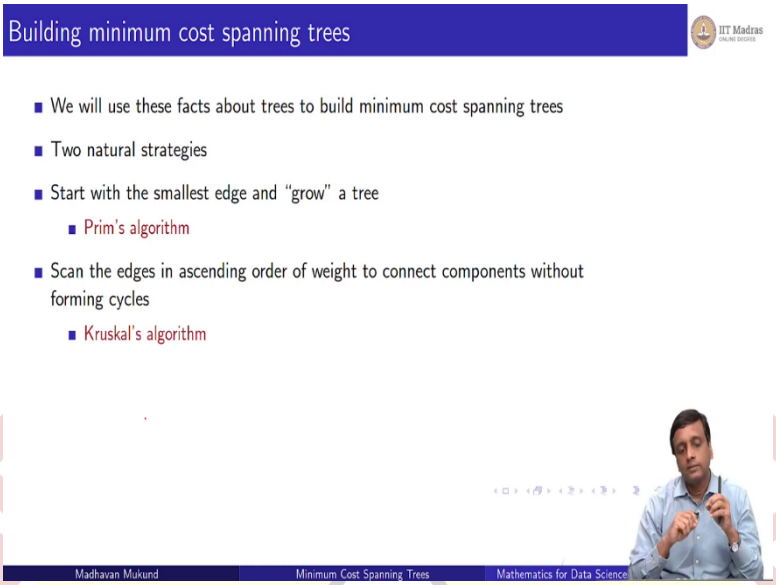
But whichever way if I can go to i one way and come back the other way, either on the entire full circuit or somewhere in between, there must be the cycle where I can go around the cycle. So, if I have multiple paths from i to j, there must be a cycle somewhere. So, these are these 3 facts about trees. So, it has exactly n-1 vertices. If I add an edge, it creates a cycle, and there is a unique path between any two vertices.

(Refer Slide Time: 11:21)



So, to combine this, we can say that, if I give you any two of these conditions, then the graph is a tree. So, if I tell you that the graph is connected, and acyclic, what is the definition of tree, I showed you that it has n-1 edges, it is connected as n-1 edges, then it must be acyclic, if it is acyclic and n-1 edges must be connected. So, if I tell you any two of these three facts, you can conclude that the graph you are looking at is a tree. So, this is a very useful thing to remember when you are going forward.

(Refer Slide Time: 11:49)



So, we are going to use some of these facts in order to design algorithms for this problem that we are considering, which is to build a minimum cost spanning tree. So, remember, a minimum cost spanning tree is a tree which touches every vertex of the given graph by taking a subset of edges, which covers all the vertices. And among those, you want a tree in which the sum of the edge costs that you have used to build this tree is minimum.

So, there are two strategies that one can think of to do this. And we will look at two algorithms follow these strategies. The first strategy is to start from a single vertex or a smallest single edge and grow a tree. So, you try to build a tree incrementally, you start, and then you keep building a tree. So, you start at an edge make another tree, add an edge, make a bigger tree, to add an edge and it does not make a tree you do not consider it. So, you just grow a tree, so we will look at it is called Prim's algorithm.

The other way is to take a disconnected thing and connect it into a tree. So, initially, you can say that all the vertices are apart and you say, let me take a small edge and connect to things. So, now I have got the starting point. Let me take two other edges, two other vertices, connect them and then let me connect this to that. So, you build a tree by kind of grouping together the components rather than growing one tree. So, this is called Kruskal's algorithms. We will see both of these in detail. So, you will understand the difference between these two strategies and see how they work.