INDIAN INSTITUTE OF TECHNOLOGY MADRAS

सिद्धिर्भवति कर्मजा

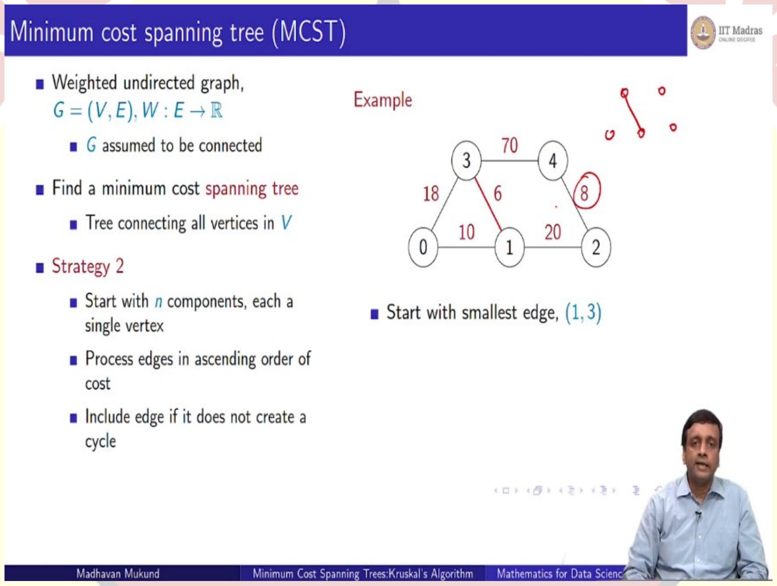
IIT Madras

ONLINE DEGREE

Mathematics for Data Science 1
Professor. Madhavan Mukund
Chennai Mathematical Institute
Lecture No. 12.7
Minimum Cost Spanning Tress: Kruskal's Algorithm

We are looking at minimum cost spanning tress and we saw one strategy the Prim's strategy which tries to incrementally grow a tree starting with one vertex or one edge until you get an overall tree which is minimum cost. The other strategy is called Kruskal's Algorithm.

(Refer Slide Time: 0:32)



Minimum cost spanning tree (MCST)

- Weighted undirected graph, $G = (V, E), W : E \rightarrow \mathbb{R}$
  - G assumed to be connected
- Find a minimum cost spanning tree
  - Tree connecting all vertices in V
- Strategy 2
  - Start with n components, each a single vertex
  - Process edges in ascending order of cost
  - Include edge if it does not create a cycle

Example

- Start with smallest edge, $(1, 3)$

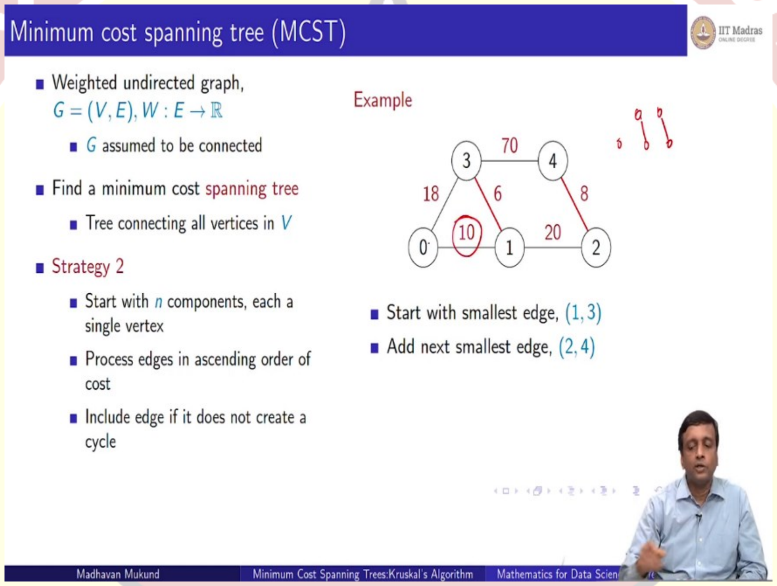
IIT Madras
Madhavan Mukund
Minimum Cost Spanning Trees:Kruskal's Algorithm
Mathematics for Data Scienc

So, remember that we have working with weighted undirected graphs. So, we have a weight function associating a number with every edge. And we assume the graph is connected. And we want to find the minimum cost spanning tree which connects all the vertices in V. So, in Kruskal's Algorithm what we do is we start with all the vertices disconnected forming n components. And then we try to merge components. We try to connect components by the smallest edge that we have which connects to components 2.

So, let us do an example and then we will do it in more detail. So, this our familiar example so here the first thing that we do in Kruskal's Algorithm is to sort the edges in ascending order. So, we will sort the edges in ascending order and then we start with the smallest edge. So, the smallest edge is in this case. So, initially we are imagining that we have this disconnected graph consisting of these five vertices which are just siting an isolation. Then we bring this one edge in and we will create this component.

So, now we have 4 components one component has the vertices 1 and 3. And the other three are these isolate components. Now in Prim's Algorithm you would take this component and you would extend the tree. In Kruskal's Algorithm you just take the smallest edge which connects two components and makes them into a larger component. So, in this case I jump from 6 to 8. That is the next smallest edge and it is connecting two components which are separate at the moment.

(Refer Slide Time: 1:57)



Minimum cost spanning tree (MCST)

- Weighted undirected graph, $G = (V, E), W : E \to \mathbb{R}$
  - G assumed to be connected
- Find a minimum cost spanning tree
  - Tree connecting all vertices in V
- Strategy 2
  - Start with n components, each a single vertex
  - Process edges in ascending order of cost
  - Include edge if it does not create a cycle

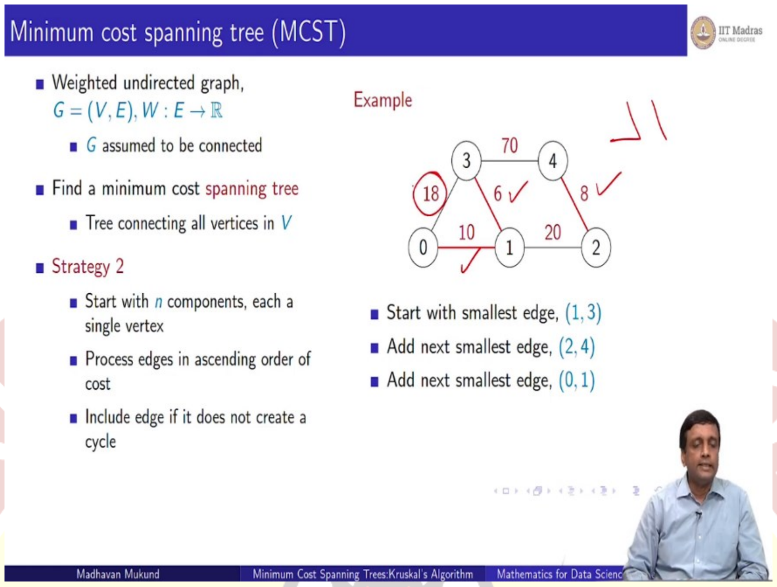
Example

- Start with smallest edge, $(1, 3)$
- Add next smallest edge, $(2, 4)$

IIT Madras
Madhavan Mukund
Minimum Cost Spanning Trees:Kruskal's Algorithm
Mathematics for Data Scien

So, I am allowed to add that and I get now these two components I have three components, I have now constructed this kind of a graph. Where I have 0 separate 1 and 3 and then 2 and 4. Now what is the next one? The next one is this 10, so it will connect 0 to 1.

(Refer Slide Time: 2:16)



Minimum cost spanning tree (MCST)

- Weighted undirected graph, $G = (V, E)$, $W : E \to \mathbb{R}$
  - G assumed to be connected
- Find a minimum cost spanning tree
  - Tree connecting all vertices in V
- Strategy 2
  - Start with n components, each a single vertex
  - Process edges in ascending order of cost
  - Include edge if it does not create a cycle

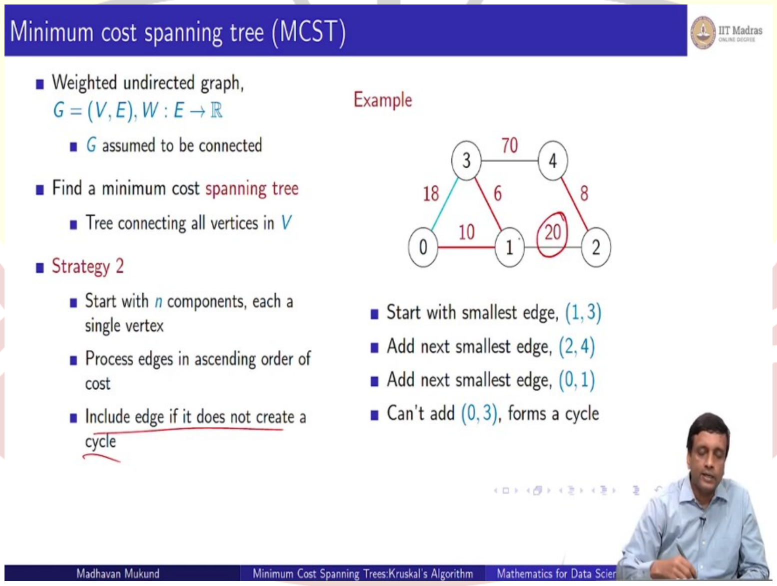
Example

- Start with smallest edge, $(1, 3)$
- Add next smallest edge, $(2, 4)$
- Add next smallest edge, $(0, 1)$

3
70
4
18
6
8
0
10
1
20
2
Madhavan Mukund
Minimum Cost Spanning Trees:Kruskal's Algorithm
Mathematics for Data Science

So, I am allowed to add it. So, now I have this kind of situation, so I still have overall disconnected graph it is not a tree yet. And now I have to decide what to do. So, I first is 6, 8, and 10. So, the next vertex in ascending order of cost is 18.

(Refer Slide Time: 2:38)



Minimum cost spanning tree (MCST)

- Weighted undirected graph, $G = (V, E)$, $W : E \to \mathbb{R}$
  - G assumed to be connected
- Find a minimum cost spanning tree
  - Tree connecting all vertices in V
- Strategy 2
  - Start with n components, each a single vertex
  - Process edges in ascending order of cost
  - Include edge if it does not create a cycle

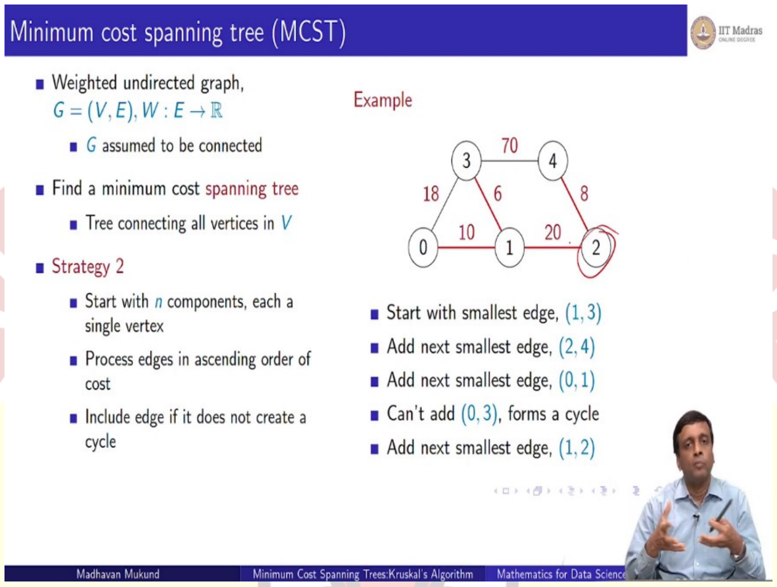
Example

- Start with smallest edge, $(1, 3)$
- Add next smallest edge, $(2, 4)$
- Add next smallest edge, $(0, 1)$
- Can't add $(0, 3)$, forms a cycle

3
70
4
18
6
8
0
10
1
20
2
Madhavan Mukund
Minimum Cost Spanning Trees:Kruskal's Algorithm
Mathematics for Data Science

Who I discover now is at when I try to add 18 it does not connect two different components. It connects two vertices in the same component. So, it forms a cycle and this is not good. So, Kruskal's Algorithm says include an edge if it does not create a cycle. So, 18 gets discarded. So, we do not take the edge 0, 3 and we proceed to the next stage. So, the next stage in ascending order is 20.

(Refer Slide Time: 3:01)



Minimum cost spanning tree (MCST)

- Weighted undirected graph, $G = (V, E), W : E \to \mathbb{R}$
  - G assumed to be connected
- Find a minimum cost spanning tree
  - Tree connecting all vertices in V
- Strategy 2
  - Start with n components, each a single vertex
  - Process edges in ascending order of cost
  - Include edge if it does not create a cycle

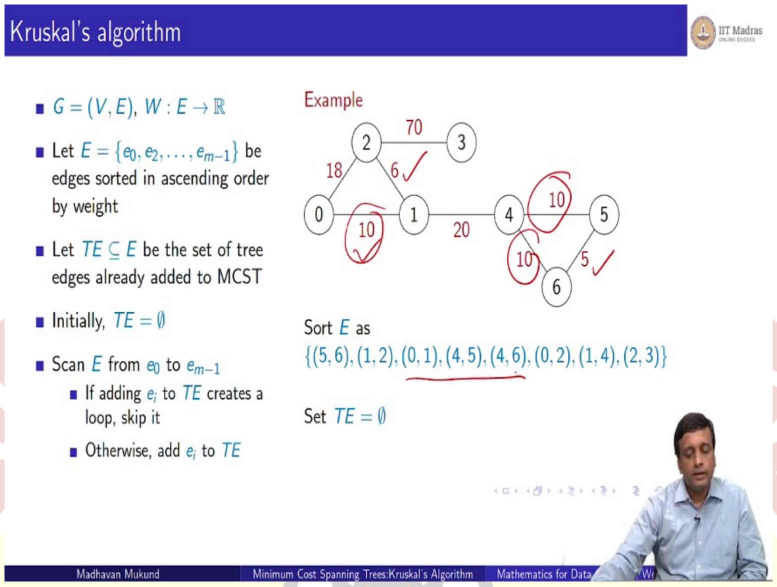
Example

- Start with smallest edge, $(1,3)$
- Add next smallest edge, $(2,4)$
- Add next smallest edge, $(0,1)$
- Can't add $(0,3)$, forms a cycle
- Add next smallest edge, $(1,2)$

IIT Madras
0
1
2
3
4
18
70
6
8
10
20
Madhavan Mukund
Minimum Cost Spanning Trees:Kruskal's Algorithm
Mathematics for Data Science

So, we add that and we have found essentially the same spanning tree we found from Prim's Algorithm but in a different sequence. Remember in Prim's Algorithm we first added 6 then we added 10 then we added 20 and finally we added 8 because we could only add 8 when we reached one end point of 8 that namely this vertex 2. Where as in Kruskal's Algorithm we added 8 upfront and we create these disjoint components. And as we go along this components kind of merge together and they form a tree.

(Refer Slide Time: 3:28)



Kruskal's algorithm

- $G = (V, E)$, $W : E \to \mathbb{R}$
- Let $E = \{e_0, e_2, \ldots, e_{m-1}\}$ be edges sorted in ascending order by weight
- Let $TE \subseteq E$ be the set of tree edges already added to MCST
- Initially, $TE = \emptyset$
- Scan E from e_0 to e_{m-1}
  - If adding e_j to TE creates a loop, skip it
  - Otherwise, add e_j to TE

Example

Sort E as $\{(5,6), (1,2), (0,1), (4,5), (4,6), (0,2), (1,4), (2,3)\}$

Set $TE = \emptyset$

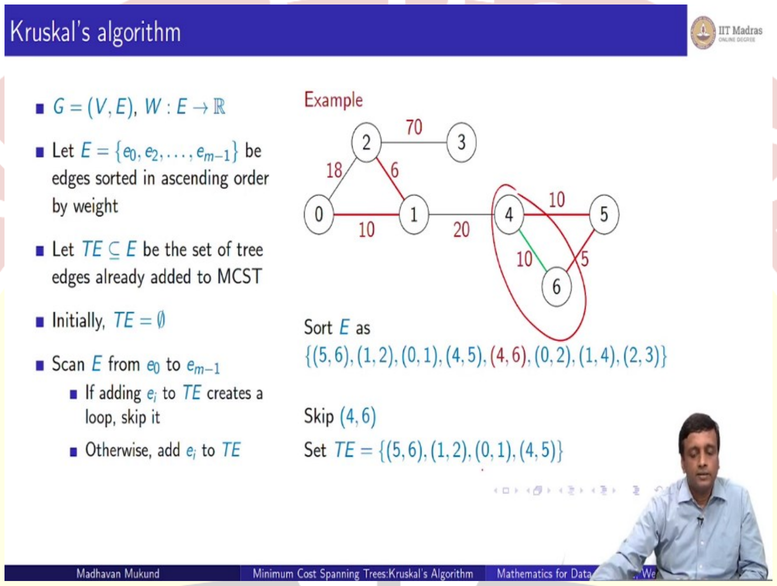
So, here is the little bit more formal definition of Kruskal's Algorithm. So, we have assume that our edges are arranged in this ascending order. So, e0 is the smallest edge this is e1, e0, e1 upto em-1. So, I sorted in ascending order by weight. And now what you do is you keep track of like we did not Prim's Algorithm. We keep track of the set of edges that we have added. And implicitly the set of edges also tells us what are the components.

So, you can if you are trying to actually write this as code you will actually keep track for the components also which is a little bit TDS when you are programming this. But mathematically we know the edges then we can compute the components by just doing kind of the reachability on each component. And finding out which components are. So, initially the set of edge is empty. And now we scan all the edges from the smallest edge to the largest edge.

And if adding it creates a loop or a cycle we skip it otherwise we add it. So, here is the little bit more complicated graph to try and see how this works. So, these are my 0 to 6 or 7 vertices and I have some 8 edges in them. So, I sort the edges so this is my smallest edge and this is my second smallest edge and so on and this is my third. Now I have three edges which are of equal size. So, these 3 edges, so I fix some ordering we discuss that if you have equal weight edges you just fix some ordering on them.

So, I have fixed some ordering which is basically based the lexicographic ordering of the end points. So, 0, 1 comes before 4, 5 because 0 comes in 4, 4 and 4, 5 comes before 4, 6 because 5 come before 6. So, I have just chosen this ordering you can choose any ordering. So, you fix an order in which you are going to process the vertices. Such that this an ascending order if the equal vertices you choose some way to group them so that they are in some fixed order. So, initially, now my edge set is empty and I pick the smallest one.

(Refer Slide Time: 5:21)



Kruskal's algorithm

- $G = (V, E)$, $W : E \to \mathbb{R}$
- Let $E = \{e_0, e_2, \ldots, e_{m-1}\}$ be edges sorted in ascending order by weight
- Let $TE \subseteq E$ be the set of tree edges already added to MCST
- Initially, $TE = \emptyset$
- Scan E from e_0 to e_{m-1}
  - If adding e_i to TE creates a loop, skip it
  - Otherwise, add e_i to TE

Example

Sort E as $\{(5,6),(1,2),(0,1),(4,5),(4,6),(0,2),(1,4),(2,3)\}$

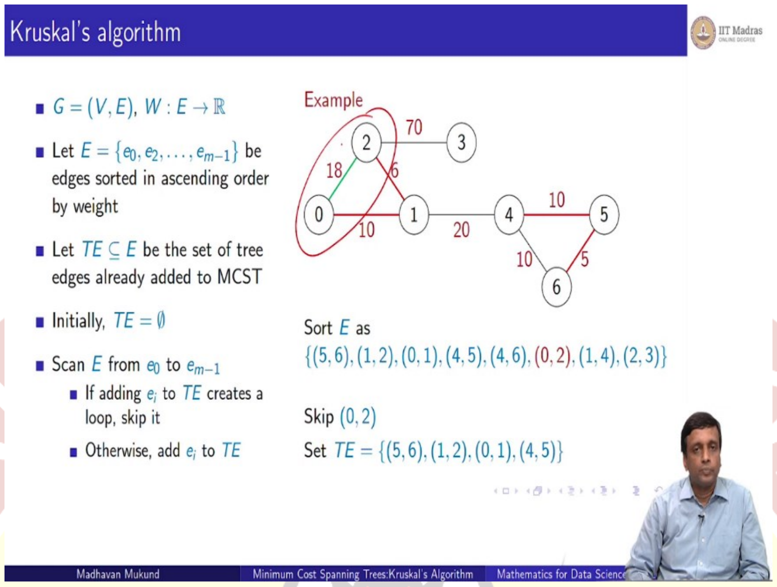
Skip $(4,6)$

Set $TE = \{(5,6),(1,2),(0,1),(4,5)\}$

So, now I process it from smallest to largest. So, I pick the smallest one no problem it does not create any I have got no components before this. So, I now created one component, so I can add it and I have 5, 6 as the tree edge. Now I look at the next stage so remember this is useful that I have already sorted it. So, I know the next stage I have to process is 1, 2. So, I look at 1, 2 again no problem it connects two different components so I am fine. Then I look at 0, 1. 0, 1 also connects two different components so I am fine.

So, now I have come to the 10 block. So, now I look at 4, 5. 4, 5 also connects two different things because 5, 6 was m component and 4 will m component I am fine. Now I come to 4, 6 and 4, 6 now is actually lying within a component. So, I cannot use 4, 6, 5 skip it. So, I do not add to the edge set. So, I leave the edge set unchanged.

(Refer Slide Time: 6:17)



Kruskal's algorithm

- $G = (V, E)$, $W : E \to \mathbb{R}$
- Let $E = \{e_0, e_2, \ldots, e_{m-1}\}$ be edges sorted in ascending order by weight
- Let $TE \subseteq E$ be the set of tree edges already added to MCST
- Initially, $TE = \emptyset$
- Scan E from e_0 to e_{m-1}
  - If adding e_i to TE creates a loop, skip it
  - Otherwise, add e_i to TE

Example

Sort E as
$\{(5,6), (1,2), (0,1), (4,5), (4,6), (0,2), (1,4), (2,3)\}$

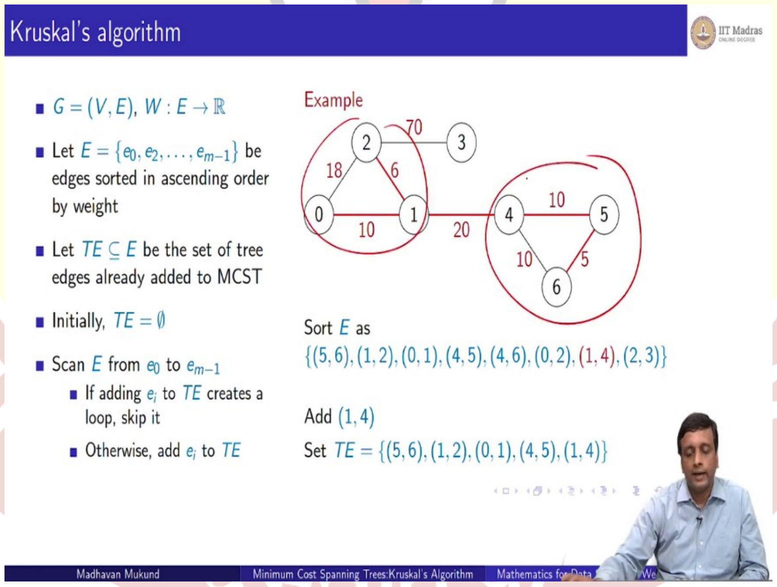
Skip $(0,2)$

Set $TE = \{(5,6), (1,2), (0,1), (4,5)\}$

Madhavan Mukund Minimum Cost Spanning Trees:Kruskal's Algorithm Mathematics for Data Science

Now I come to 0, 2 same problem 0, 2 is connecting two vertices which are already in the dame component. So, I have to skip that also.

(Refer Slide Time: 6:27)



Kruskal's algorithm

- $G = (V, E)$, $W : E \to \mathbb{R}$
- Let $E = \{e_0, e_2, \ldots, e_{m-1}\}$ be edges sorted in ascending order by weight
- Let $TE \subseteq E$ be the set of tree edges already added to MCST
- Initially, $TE = \emptyset$
- Scan E from e_0 to e_{m-1}
  - If adding e_i to TE creates a loop, skip it
  - Otherwise, add e_i to TE

Example

Sort E as
$\{(5,6), (1,2), (0,1), (4,5), (4,6), (0,2), (1,4), (2,3)\}$

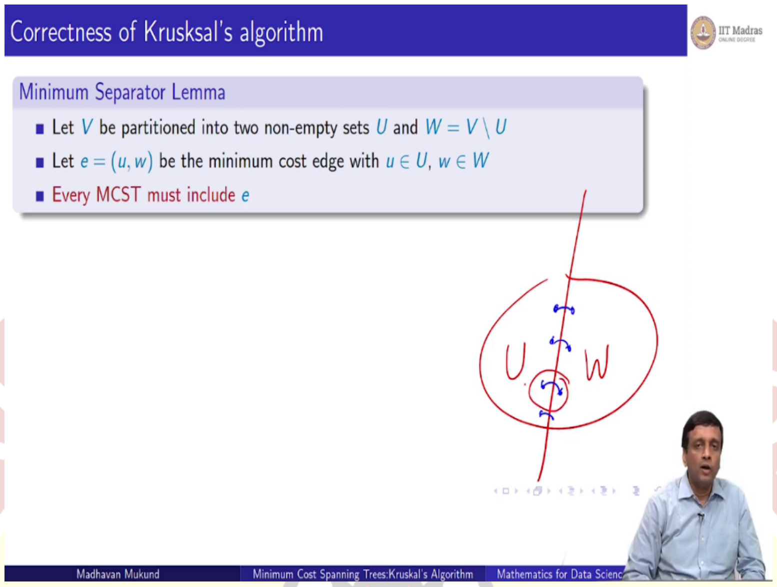
Add $(1,4)$

Set $TE = \{(5,6), (1,2), (0,1), (4,5), (1,4)\}$

Madhavan Mukund Minimum Cost Spanning Trees:Kruskal's Algorithm Mathematics for Data

So, now after 18 I go to 20. And that is not a problem because it takes this component and this component and connects them they are two different components. So, I add that and finally that edge 70 which connects 2 to 3 is added. So, this is how Kruskal's Algorithm works.

(Refer Slide Time: 6:40)



Correctness of Krusksal's algorithm

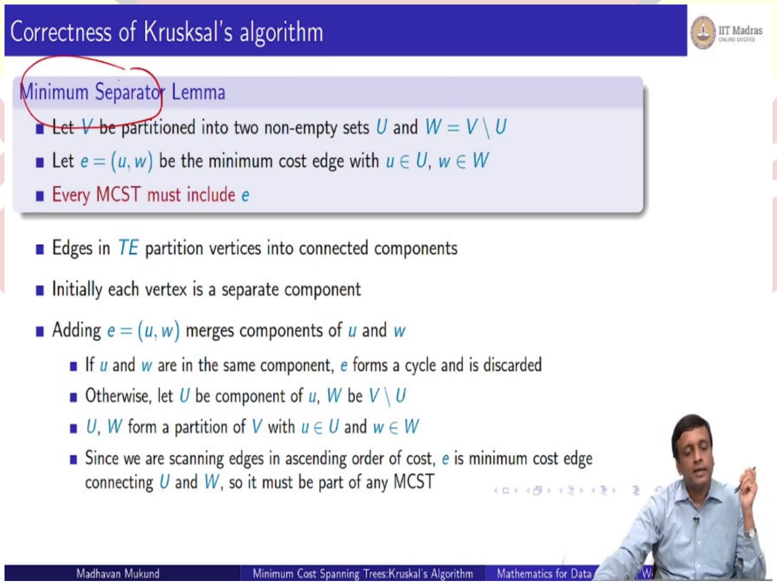
Minimum Separator Lemma

- Let V be partitioned into two non-empty sets U and $W = V \setminus U$
- Let $e = (u, w)$ be the minimum cost edge with $u \in U$, $w \in W$
- Every MCST must include e

U
W
Madhavan Mukund
Minimum Cost Spanning Trees:Kruskal's Algorithm
Mathematics for Data Scienc

So, the reason that Kruskal's Algorithm is correct is exactly the same reason that Prim's Algorithm is correct is because of this minimum separator lemma remember what the minimum separator lemma says if I take my graph and I split it so that there are some vertices on this side and some on the other side. There are this partition into U and W no matter how I partition it. If none sit and look at all the edges which cross and then among these. If I pick the smallest one then that smallest one must belong to every MCST.

(Refer Slide Time: 7:14)



Correctness of Krusksal's algorithm

Minimum Separator Lemma

- Let V be partitioned into two non-empty sets U and $W = V \setminus U$
- Let $e = (u, w)$ be the minimum cost edge with $u \in U$, $w \in W$
- Every MCST must include e

- Edges in TE partition vertices into connected components
- Initially each vertex is a separate component
- Adding $e = (u, w)$ merges components of u and w
  - If u and w are in the same component, e forms a cycle and is discarded
  - Otherwise, let U be component of u, W be $V \setminus U$
  - U, W form a partition of V with $u \in U$ and $w \in W$
  - Since we are scanning edges in ascending order of cost, e is minimum cost edge connecting U and W, so it must be part of any MCST

So, in our think the edges that we have found so far in Kruskal's Algorithm partition the vertices into connected components. So, now initially each vertex is in a separate component. And when I add an edge u, w it merges the components of u and w. So, basically the thing is that if I am keeping track of which vertices are in the same component and I can do that incrementally. Even I add an edge I can grow the component I can say the component containing u now contains w also. So, it is not a problem.

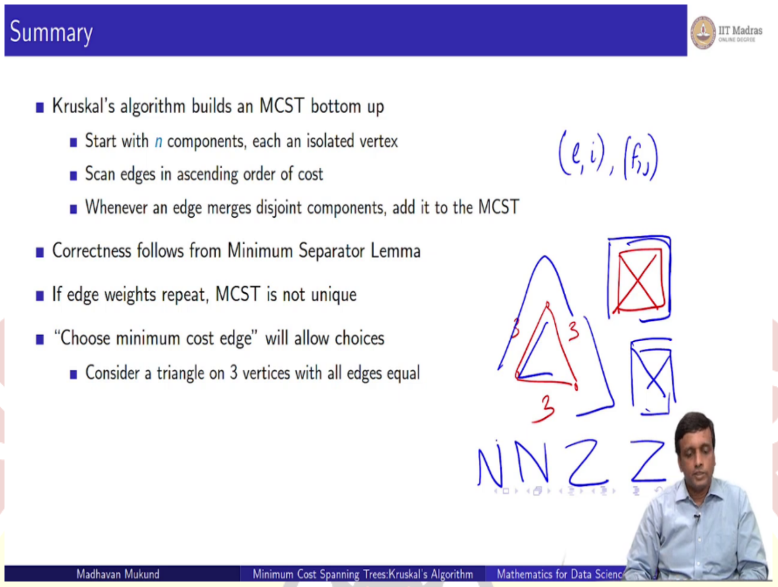
So, I do this as I go along, so if I discover that the edge I want to add actually falls within the component both end points is the same component. It will then form a cycle and then I discover it. On the other hand if it connects two different components then we can apply this lemma to argue that what we are doing is correct because we look at the component containing the starting point. So, let capital U be the component containing small u. And let W be the rest.

Now W has because we are assuming that U and W are in u small u and small w are different components, I have U which contains the small u and this W which is the rest it contains small w. Now we are processing these edges in ascending order. So, since these are in different components I have not yet connected these two. So, I have not found any edges yet between 0 between capital U and capital W.

Because if I have found them I would have already connected these components. So, the reason they are not connected is because I have not found them yet. But among the edges which I have not looked at the edge I am looking at right now is the smallest one. Because I am doing it in ascending order. So, this forms a partition and we have forming this we are scanning in ascending order.

So, this edge that I am looking at must be the smallest edge connecting capital U to capital W. So, it is satisfies this property of this minimum separator. It is a minimum separator of capital U and capital W. So, what Kruskal's Algorithm would do? (())(9:15) pick it up an edit. And it is correct because its separator says that any such edge which is the minimum separator between this partition and that partition. And must be there in every MCST. So, Kruskal's Algorithm is correct for the same reason that Prim's Algorithm is correct because of this minimum separator lemma.

(Refer Slide Time: 9:33)



Summary

- Kruskal's algorithm builds an MCST bottom up
  - Start with n components, each an isolated vertex
  - Scan edges in ascending order of cost
  - Whenever an edge merges disjoint components, add it to the MCST
- Correctness follows from Minimum Separator Lemma
- If edge weights repeat, MCST is not unique
- "Choose minimum cost edge" will allow choices
  - Consider a triangle on 3 vertices with all edges equal

(e,i), (f,j)
3
3
3
N N Z Z
Madhavan Mukund
Minimum Cost Spanning Trees:Kruskal's Algorithm
Mathematics for Data Science

So, the difference between Prim's Algorithm and Kruskal's Algorithm is basically Kruskal's Algorithm is kind of assembles the tree bottom up. So, it takes all this just the connected things and then it goes around putting them together. Whereas, the Prim's Algorithm starts somewhere and it grows a tree gradually to cover the whole thing. So, they have different strategies but both of them owe that correctness to that one lemma which says that whenever I partition the vertices into two disjoint sets.

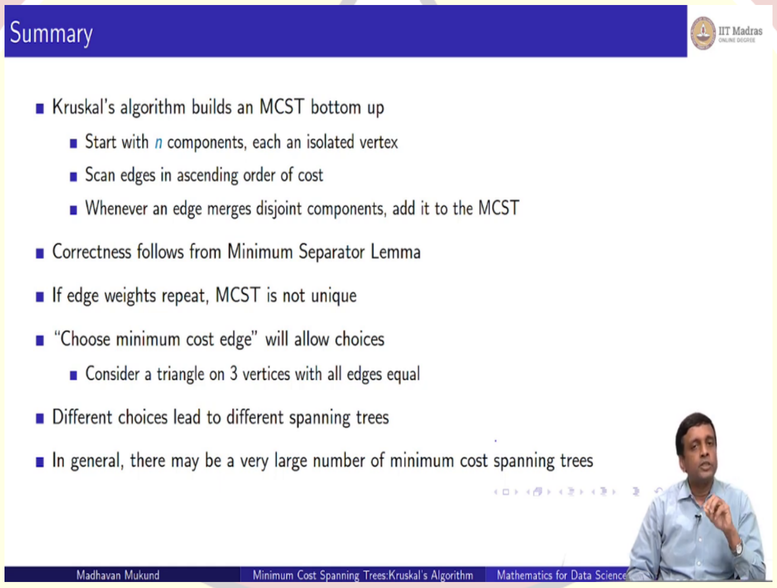
The smallest edge connecting these two partitions must be there in every MCST. Now if there are repeated edge weights then we already saw in the unweighted case that there are many spanning tress. If there are repeated the same weight repeats, then we may not get a unique spanning tree. So, for instance supposing I take a very simple graph which is just this 3 vertex graph and I put weight 3 here, here and here.

Then, any of these would be a spanning tree that I could take this pair of edges or I could take this pair of edges or I could make this pair of edges. So, all of them would be minimum cost spanning tress. And where it comes in our algorithm means when we say choose the minimum cost edge remember we have said that we might actually have to specify it as some f comma j and so on.

So, this ordering that we choose will decide which one will get picked up. So, that is why we get different choices. So, here you can see that this triangle on four vertices. And three vertices has actually three spanning tress. If I do a more complicated thing for instance supposing I take a square. Pick the four two diagonals then what are the spanning tress well the spanning tress. So, there are some obvious tress like this one. So, this is a spanning tree. When I take 3 edges around 4 vertices or spanning tree will have 3 edges.

But these are also spanning tress. Two edges and a diagonal this z is also a spanning tree. Two edges and diagonal connecting them. How many of them are there well there are 4 ways of going around the outside I can have this, I can have this, I can have this. Then there are some 4 ways of choosing the corner to include and the then the z can also be in many ways. So, it is many different orientations. So you can see that with 6 edges I can get a anonymous number of spanning.

(Refer Slide Time: 11:48)



Summary

- Kruskal's algorithm builds an MCST bottom up
  - Start with n components, each an isolated vertex
  - Scan edges in ascending order of cost
  - Whenever an edge merges disjoint components, add it to the MCST
- Correctness follows from Minimum Separator Lemma
- If edge weights repeat, MCST is not unique
- "Choose minimum cost edge" will allow choices
  - Consider a triangle on 3 vertices with all edges equal
- Different choices lead to different spanning trees
- In general, there may be a very large number of minimum cost spanning trees

Madhavan Mukund | Minimum Cost Spanning Trees:Kruskal's Algorithm | Mathematics for Data Science

So, in general different choices lead to different spanning trees. And there are not unique edge weight, then I could get a very large number of spanning. So, depending how I have chosen to order this equal edge weights. Prim's Algorithm and Kruskal's Algorithm will pick one particular one out of these. It will not give you all of them it will give you one of them. And if they are disjoint in the sense that it not disjoint if they are distinct that is if all the edge weights are different then using the minimum separator lemma you can argue that every choice on Prim’s Algorithm is forced. Every choice in Kruskal's Algorithm is forced and they will give you exactly the same spanning trees.

So, as long as the edge weights are disjoint it does not matter whether use Prim's or Kruskal's you will get the same spanning tree. But if the edge weights repeat then depending on how you choose to order the vertices. The two algorithm might produce different spanning trees with the same minimum cost but different set of edges. So, keep that in mind.