Data Structures and Algorithms
Dr. Naveen Garg
Department of Computer Science and Engineering

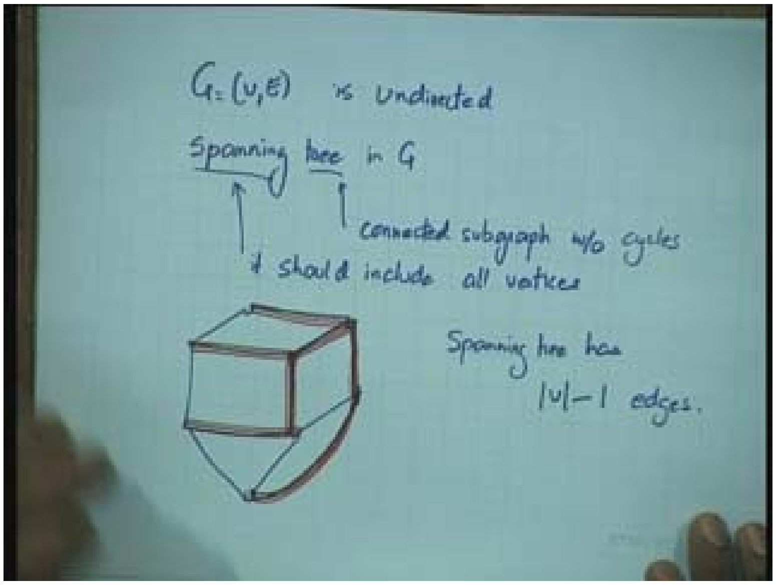
Indian Institute of Technology, Delhi
Lecture – 31

Today we are going to be talking about minimum spanning trees. We are going to define what a minimum spanning tree is. We are also going to look at algorithms of minimum spanning trees. So you all know what a spanning tree is. Does everyone know what a spanning tree in a graph is? Did we define it? Yes we did. So I gave you an undirected graph, so what is a spanning tree in G?

So spanning tree, the term is composed of two things. Tree and panning, tree means it should be a tree. What is a tree? Tree is a connected sub graph without any cycles. Connected sub graph without cycles that is a tree. And what does spanning mean? Spanning means that it should include all vertices. This basically means that it should include all vertices that would be a spanning tree in the graph.

So for an example if I drew a graph something like that, say this were my graph. Then I could draw a spanning tree in this graph by let say, I am going to use a very, let us say, I pick this edge in the tree, pick this, pick that, pick that then I pick this and I pick this. How many edges do I need to pick? Is this a spanning tree? No. This vertex is not included here. I could pick one of the edges incident at this to include this. This is a spanning tree. We know that our spanning tree, if the graph has n vertices, so spanning tree has how many edges? number of vertices minus 1 edge. Yes: great.

(Refer Slide Time: 03:43)



$G = (V, E)$ is undirected

Spanning tree in G

connected subgraph w/o cycles

it should include all vertices

Spanning tree has $|V| - 1$ edges.

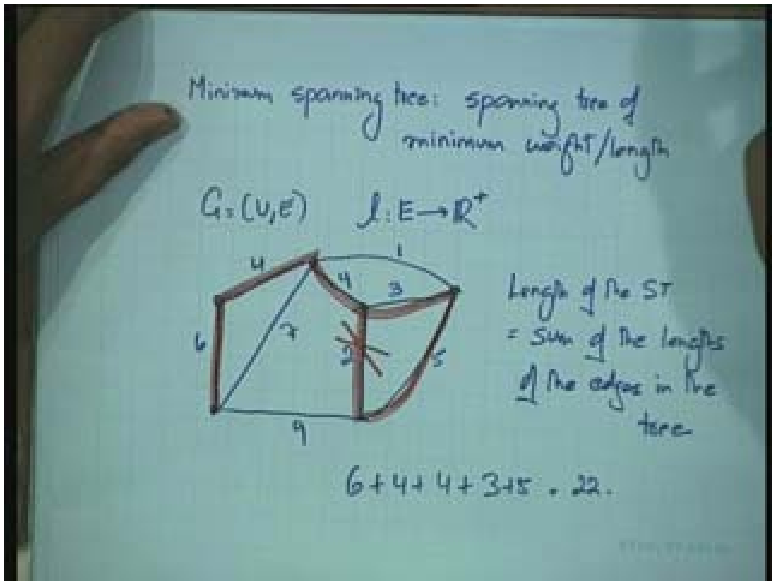
What is a minimum spanning tree? Minimum spanning tree is a spanning tree of minimum weight. Sometimes I will use the term weight, sometimes I will use the terms length, one at the same thing. (Hindi) you are not just given the graph but we are also given a length function on the edges of the graph. Let me assume the lengths are non-negative reals. So, every edge has a length now. So, think of that in the following way. These are certain cities and I can either draw or I can either connect these 2 cities by a wire or I can connect these 2 cities by a wire or I can connect these 2 by a wire, I can connect these, these or these.

These are the possible options I have of connecting them with wires. Some options are not there. Why? (hindi) could be some reason (hindi). So, this is a graph. Now, to connect 2 particular cities, I am also told, how much length of the wire I need, that is my length function. Suppose I told you that you need 3 here and you need 5 here and you need 2 here, what are the units? It could be anything, 1000 kilo meters, meters, I do not care. This could be 1, this could be 7, this could be 4, 6, 9.

I just put in some numbers in between. So, we need to connect these cities which means we need to create a spanning tree and I am not interested in any spanning tree. I am interested in spanning tree for which the length of wire is spent as small as possible. So, that is what I mean by a minimum spanning tree, a spanning tree of a minimum length. What is the length of the spanning tree now? So, sum of the lengths of the edges in the tree. So, let us formally define that the length of the spanning tree equals sum of the lengths of the edges in the tree.

So, I might now decide to pick some edges. Suppose I pick, I did not give this length, I give this the length 4. Suppose, I pick this edge and I pick this edge and I pick this edge and I pick this edge and I pick this edge. (hindi) So, I cannot pick this edge, it is not a tree at all, I cannot pick. So, let me pick something else. So, let me pick this 6. (hindi) What is the length of this tree? 6 plus 4 plus 4 plus 3 plus 5, 22 right? There might be other trees in this, other spanning trees which are smaller than 22. I am interested in the finding the one which is the smallest.

(Refer Slide Time: 07:46)



Minimum spanning tree: spanning tree of minimum weight/length

$G=(V,E)$ $l: E \rightarrow R^{+}$

Length of the ST = Sum of the lengths of the edges in the tree

$6+4+4+3+5 = 22.$

There is no notion of the root. In the case of spanning tree, when we defined, spanning tree is trees in graph, there was no notion of the root over. We call that also, we use the term free tree for that. There is no notion. Since, the vertex, since all the vertices are going to be included (hindi) 3, 4, 1, 2, 5, 9, 7, 4, 6 and the edges we included were, this edge, (hindi) one way you can convince yourself that this is not the smallest possible tree is, by seeing that for instance.

If I include this edge of the length 1 then what is going to happen? Will it be a tree or … We will be have a cycle. So, that is one property of a spanning tree always (hindi). It is a connected sub graph which means (hindi). So, I can go from, let say, this is vertex U and this is vertex V, I can go from U to V by following the edges of the tree and then I can take this edge that is added just now, back to go to U which means it form a cycle.

If it is a cycle, it is not a tree any more. Now, in this cycle, suppose the edges have a certain length (hindi). Now, what can I do? I can include this edge and drop the edge of length 4. Will that remain? Will that continue to be a tree? If I were to drop this edge, will it remain a tree? Why? Why would it remain connected? No notion of a descendant. There is no notion of a descendant. Why? That is the question. We have discussed this before.

The point is that, if there is a path which gets it straight then that path must be using this edge. (hindi) I can now create alternate path between those 2 vertices. By instead now, going like this between this 2 edges. So, that means they are still, is the path between those 2 vertices (hindi). All paths are still there or not paths are still there, all face of vertices are still connected. May be, by different path but they are still connected.

So, that is an operation we can do always. I can try and add an edge, try to add an edge in to this tree and see if as a result of that operation, so, as a result of that operation, I will clearly form a cycle. If from that cycle, I can drop and edge of longer length than the edge I added, then I would have reduced the cost of this tree, the cost of the length or weight, I will use these terms (hindi)

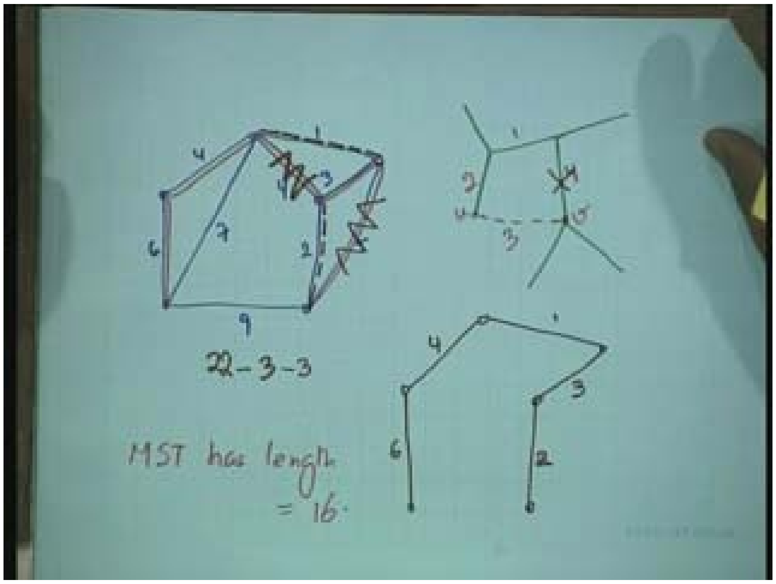
So, I form of this cycle. I can drop any edge of this cycle and it will still remain a tree. Which is the edge I would like to drop? 4, I will drop this edge, 4 and now, my new tree has length what ? (hindi), 22 minus (hindi) So, this new tree length 22 minus 3 (hindi) See, I can repeat the process, I can see, there is something else that can be done. Can you say something else that can be done? 2, so suppose, I include 2.

So, now I form the cycle which has edges of length 3 and 5 in it. So, it make sense to drop 5 and the reduction now is 3 units again. Now, let me draw the tree, we have the (hindi) So, the tree that I have at this point is, this is the tree I have now. This is the edge, length 1, this is 3, this is 2, this is 4, this is 6.

Would it make sense to include some other edge and try and repeat the process? No, because, if I include this edge which is of length 4, then this edge is the longest edge on the cycle that gets … (hindi) and the edge I am including is the longest edge on the cycle. It does not make sense to include it, therefore. If I include this edge, it is of length 5. The cycle found may, this is again the longest edge of the cycle. Does not make sense to include, If I include this edge 7, this is the cycle that get formed and 7 is again the longest edge of the cycle.

If I include this edge 9 then there is the longer cycle that gets formed. But once again 9 is the longest edge of the cycle. So, it does not make sense to include 9. So, it does not make sense to include any other edge. And so, actually, I should not say: and so, the reason is that, is not the reason why, I cannot claim from this that this is the minimum spanning tree. But, it is the minimum spanning tree. We will use different arguments proving that. So, this is a minimum spanning tree in this graph. So, MST has length, what is the length of this? 16.

(Refer Slide Time: 16:36)

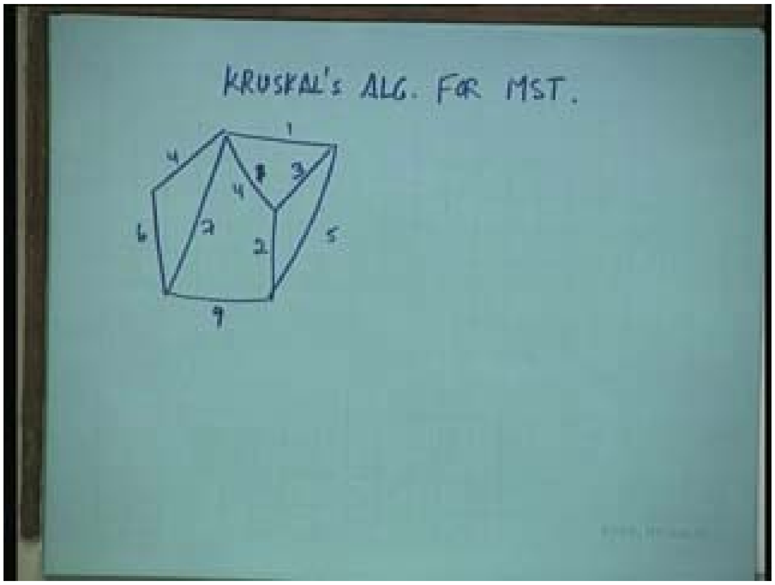


This could be treated as an algorithm, so computing the minimum spanning tree. How did it work? You start with some tree then try to include an edge, look at the cycle that gets formed, see, if you can drop an edge of longer length from that cycle or if there is a longer edge on that cycle. If there is, then you can drop that edge and therefore reduce the cost of the tree.

So, you keep doing procedure till you reduce the cost of the tree and you stop when you cannot reduce it anymore. But, we will not look at this algorithm or we will not analyse this algorithm because it will be fairly expensive, in terms of running time. We are going to look at the different algorithm for computing the minimum spanning tree and algorithm we are going to look at, is called Kruskal’s algorithm.

So, we are looking at Kruskal’s algorithm in minimum spanning tree. It is actually very simply algorithm. Let me illustrate it on the same graph that we had before. I am going to draw this graph out.

(Refer Slide Time: 18:38)



KRUSKAL's ALG. FOR MST.

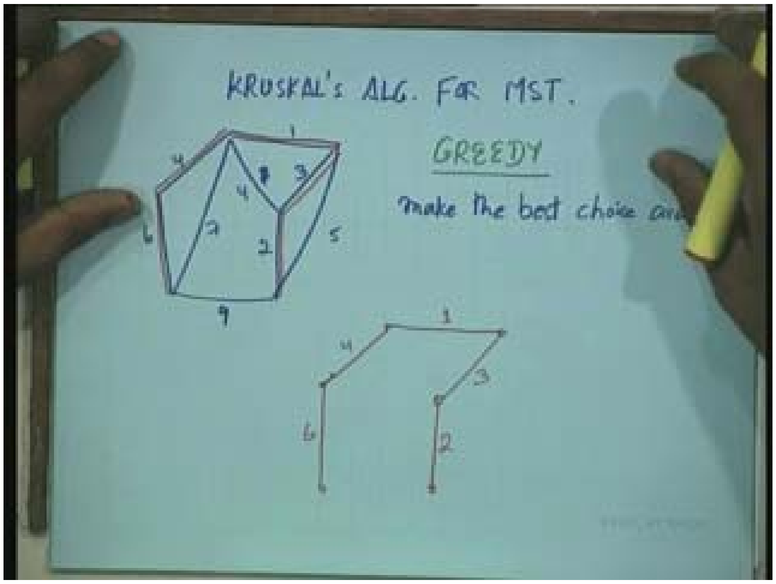
The algorithm is the following, it says, take the minimum edge in the graph. (hindi) This is an example of what is called the greedy algorithm also. You will be greedy, you are all greed, but, we will now be greedy to compute the minimum spanning tree. So, what you think greedy means? Try to get the larger. Go for less? That is not greedy. Make the best possible choice at each time without thinking about the future. That is the key thing. Make the best choice available, I will not write without thinking of the future. But that is reason why it is called greedy. We just make the best available choice at each time. Here, of course, our objective is to minimize the length of a tree.

What is the best thing to do at the first step? Take the edge with the smallest length. So, let us do that. We pick the edge with the smallest length, that is 1 and this gets included into a minimum spanning tree. So, I am now building the minimum spanning tree, edge by edge. I included the edge 1, edge of length 1. Which is the next smallest edge? Which is the next best choice to make it? 2? 2 or 3? When you are saying, we need to connect it your thinking of the future. Do not do that. Think of the, be greedy, as greedy as you can. Just take the best choice.

So, the next best is, just go for the length 2 and now what is greedy? Say next, which is the next I should take? 3, 4 (hindi) So, it will form a cycle. Since it forms the cycle, we are not getting what we wanted. It is not a tree any more. (hindi) So, you understand the only modification that I am making? If by including an edge, I form a cycle, I do not include that edge. So, I am not going to include edge 4, sorry, this, this particular edge. Which is the next edge that I would like to include? 5, I try to include 5 but 5 also forms the cycle. So, I do not include 5.

Which is the next edge I would like to include? 6, this edge, 6. Actually, I can stop at this point because now I have a spanning tree. Any other edge that I try to include will always form a cycle because this is a spanning tree and so I can stop this. As you can see, this we have already argued in, we have not argue, this is the best possible. Actually, till now, we have not seen an argument for why this is a best possible. But, this is the same solution, we obtained earlier. Soon, we are going to argue that this is the very best possible. Does every one understand? Yes? Can we write down what the algorithm is?

(Refer Slide Time: 22:42)

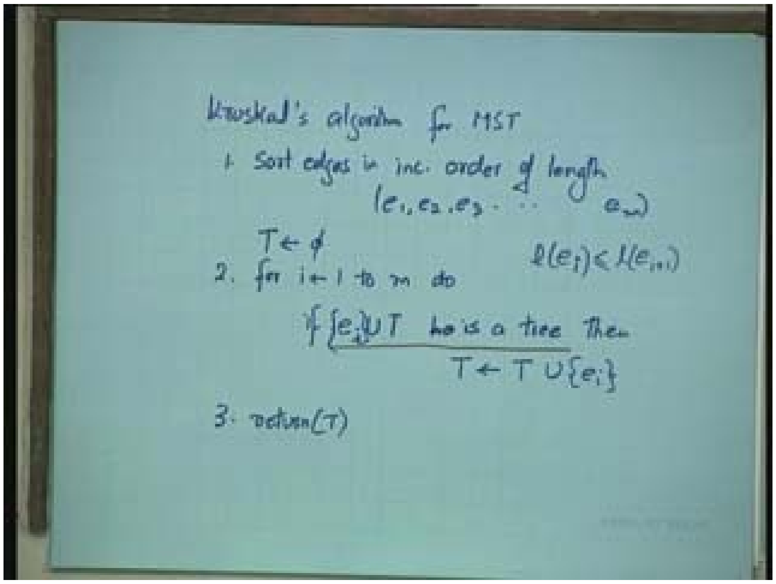


Let us quickly write down what the algorithm is, because it is very easy to write down. So, Kruskal’s algorithm (hindi). So, let us say the very first step is sort edges in increasing order of length. Let us say, this is order is e_1 e_2 e_3 $e_{m.}$ What is this mean? This means e_1 is the edge with the smallest length. So, to be more specific, l of $e_{i,}$ length of edge e_i is less than the length of the edge e_i plus 1, less than or equal to. Now, what should be the next step? For i going from 1 to m do, what should I do? Take the first element? Take e_i to that spanning tree.

So, we need to define some spanning tree. Let say, my spanning tree is T which is initially, null. T is the set of edges. So, what should I do now? If e_i union T has a cycle, or is a tree then T is T union $e_{i.}$ (hindi) return T, i plus plus for loop, (hindi) You could do that, may be. (hindi) How do we check? We can later array and we can given it a on every edges in type greater we can increases when it is greater than two that we can Wrong. (hindi)

How do you check, if that edge that you are trying to include, form the cycle or not? (hindi) What is visited means? Before including a vertex in the tree? No, we do not include the vertex in the tree. We are only including edges into the tree.

(Refer Slide Time: 26:51)



Kruskal's algorithm for MST

```
1. Sort edges in inc. order of length
        $(e_1, e_2, e_3 \ldots e_m)$
   $T \leftarrow \phi$                    $l(e_i) \leq l(e_{i+1})$
2. for $i \leftarrow 1$ to $m$ do
        if $\{e_i\} \cup T$ ho is a tree then
            $T \leftarrow T \cup \{e_i\}$
3. return$(T)$
```

As we included edges in the tree, what is visited array of that edges means? (hindi) what does visited array of vertices mean? (hindi).

(Refer Slide Time: 28:05)

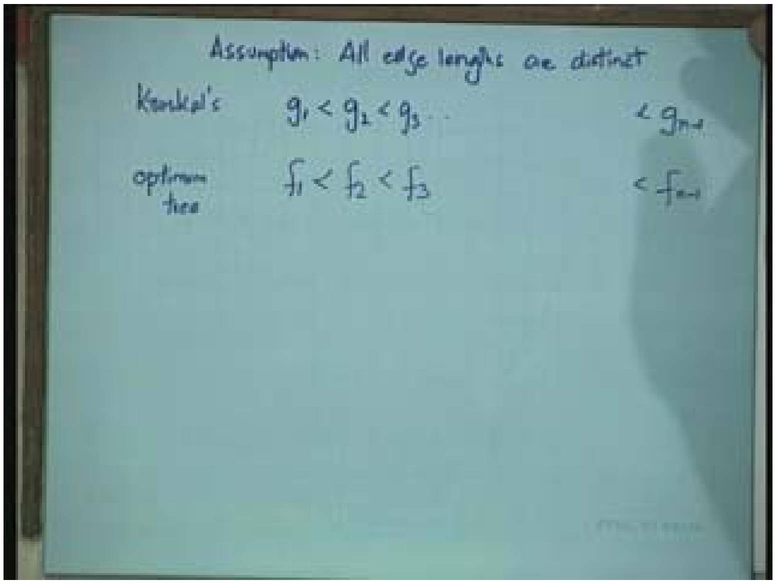
Sir, we can do that we can take that both vertices the n point of that edge and we can go for dfso and pfso DFSO, BSFO in what? We will have to develop this later. (hindi) We will see how to do it and important thing is also to do it efficiently. (hindi) because that will dictate the running time, because expect for this step, there is nothing else this algorithm. (hindi) How much time will that take? m log m (hindi) So, this will, if this step takes less then m log m time, over all iteration put together then this will be an order, m log m algorithm. (hindi) So, we have to somehow achieve that, but we will come to that point later. Before that, we have to argue, why this is a minimum spanning tree? (hindi) That means all you are saying, you know, you cannot greedily impure the tree. May be, but there is some other sequence of, (hindi) No, we are not taking always the minimum.

We are not taking always the minimum, because, we are at sometime, when the minimum forms a cycle, we through that minimum away. (conversation) So, we need to formalize this equation. All I will say is, this is not a proof. This is an intrusion for the proof. Let us see what a proof looks like. (hindi) So now, what we are going to do? Look at this carefully because this is how you will have to write your proofs in the exams. I am going to look at the edges of Kruskal's algorithm, the edges picked by Kruskal's algorithm. So, Kruskal's algorithm picks up certain set of edges. Let us call those edges, let us give those edges name.

Let us call them $g_{1,}$ g_1 is like say, the first edge in Kruskal's algorithm. So, clearly it is the edges of the, so let us g_1 g_2 $g_{3.}$ How many edges of does Kruskal's algorithm take? n minus 1,less than or equal to (hindi) all edge lengths are distinct (hindi) the same proof can be extended to a non distinct case also. That will be a question in the majors.

So, but for now, you will, we will just assume that ... The other set of edges is the set of edges that, what should be the other? So, this is the edges that Kruskal's pick. Suppose, you have figured out, what is the best possible tree is? (hindi) f_1 f_2 f_3(hindi) and again this ordering is such that f_1 is less than f_2 which is less than f_3 and so on. (hindi) This is the optimum tree. By optimum I mean, the best tree, the minimum spanning tree. Now, these 2 sets of edges need not be the same.

(Refer Slide Time: 33:37)



Assumption: All edge lenghts are distinct

Kruskal's $g_1 < g_2 < g_3 \cdots$ $< g_{n-1}$

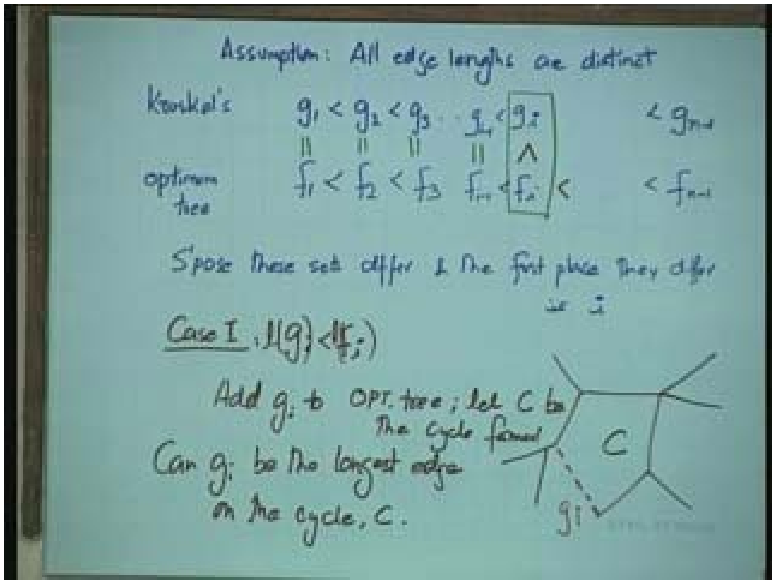
optimum tree $f_1 < f_2 < f_3$ $< f_{n-1}$

What we will argue, is that they are indeed the same. If they are the same then what we have found from Kruskal’s algorithm is the best tree. So, if these 2 set of edges are not the same, so, the proof is by contradiction. Suppose, these set differ and the first place they differ is i. What you mean by that? (hindi).Because, that will mean that g_i and f_i are also the same.

So, let us say, this is the first place where they differ, first point of difference. (hindi) Case 1, $g_{i,}$ let say less than $f_{i.}$ (hindi) every, all edge lengths are distinct. (hindi) g_i cannot be 1 of these because it is of length, strictly less than, (hindi) I actually mean, length of g_i less then length of $f_{i,}$ just to be more particular. I am not going to write it again but that is what I, it really means.

So, g_i less than $f_{i,}$ then that means, the g_i cannot be 1 of these and it cannot be 1of these. Can it one of these? No, because these are the same as these and they are all distinct. So, these cannot be the same, so that (hindi) So, g_i less than $f_{i,}$ add g_i to opt tree. (hindi) g_i is the longest edge in the cycle, can g_i be the longest edge on the cycle? Add g_i to optimum tree. Let C be the cycle formed. This cycle, I am calling it C.

(Refer Slide Time: 39:54)



Assumption: All edge lengths are distinct

$$\begin{array}{lcccccccc} \text{Kruskal's} & g_1 < & g_2 < & g_3 & g_{i-1} < & g_i & & & < g_{n-1} \\ & \| & \| & \| & \| & \wedge & & & \\ \text{optimum tree} & f_1 < & f_2 < & f_3 & f_{i-1} < & f_i & < & & < f_{n-1} \end{array}$$

S'pose these sets differ & the first place they differ is i

Case I : $l(g_i) < l(f_i)$

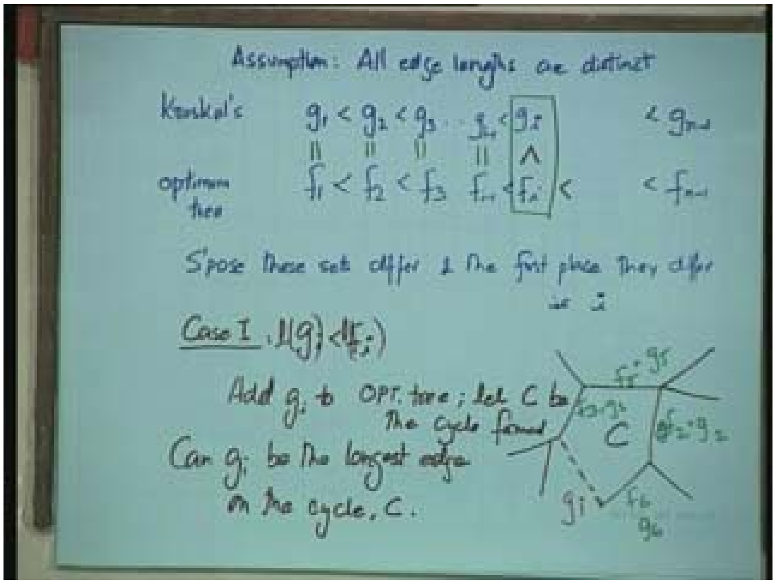
Add g_i to OPT. tree ; let C be the cycle formed

Can g_i be the longest edge on the cycle, C.

You are saying (hindi) all of these edges had length less than $g_{i,}$ what are these edges? These edges are some edges are from this side. (hindi) They are all from this set, f_1 to $f_{i,}$ n minus 1. Now, if every edge has length less than $g_{i,}$ then these edges have to be f_1 through f_i minus 1, yes.

These edges have to be from the set f_1 to f_i minus 1. (hindi) Then these edges have to be from the set, f_1 to f_i minus 1. But, f_1 through f_i minus 1 is identical to g_1 to g_i minus 1. (hindi) Contradiction, should I repeat? No?

(Refer Slide Time: 43:30)



Assumption: All edge lengths are distinct

$$\begin{array}{lcccccccccccc} \text{Kruskal's} & g_1 & < & g_2 & < & g_3 & \cdots & g_{i-1} & < & \boxed{g_i} & & < & g_{n-1} \\ & \| & & \| & & \| & & \| & & \wedge & & & \\ \text{optimum tree} & f_1 & < & f_2 & < & f_3 & & f_{i-1} & < & f_i & < & < & f_{n-1} \end{array}$$

S'pose these sets differ & the first place they differ is i

Case I. $l(g_i) < l(f_i)$

Add g_i to OPT. tree; let C be the cycle formed

Can g_i be the longest edge on the cycle, C.

$f_5 = g_5$
$f_3 = g_3$
C
$f_2 = g_2$
g_i
f_6
g_6

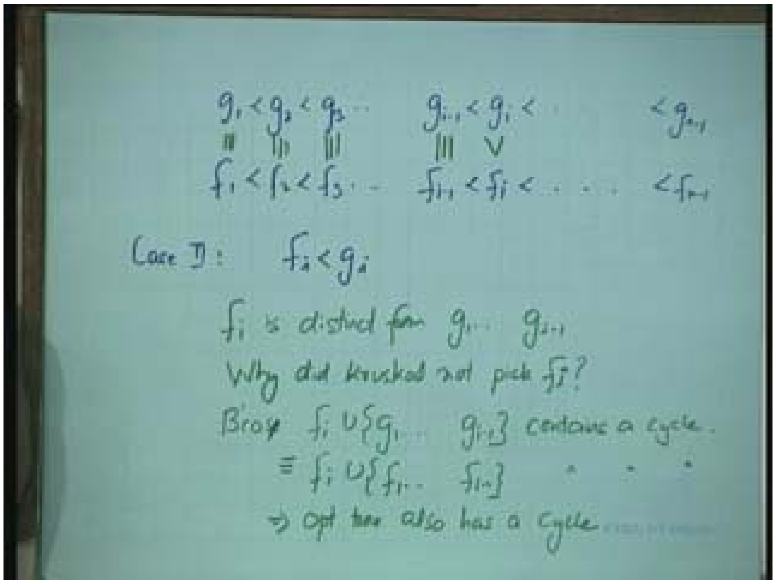
Quick repetition, g_i is less than f_i, I add g_i to the optimum tree, to this tree. (hindi) But, this set is identical to this set. Great, that is why this cannot happen.

Let us look at the other cases. Do you want me to repeat? Does everyone following the line of argument? There are 2 or 3, 2 or 3 fine point there. But, let us look at the other case. Let me 1 second, write down this, because it is useful to have this.

We are not talking of case 2 and this case is when f_i is less than g_i. Recall that these are identical and now f_i is less than g_i, that is what we are going to do now. (hindi) f_i is distinct from $g_1$to g_i minus 1. (hindi) Why did Kruskal not pick f_i? Ask him, why ask us? Why did Kruskal not pick f_i? Because they form the cycle, that was the only reason why he did not pick f_i (noise).

If it forms the cycle, so because, f_i union g_1 through g_i minus 1 contains a cycle, these set of edges (hindi) but these set of edges is identical to f_i union f_1 to f_i minus 1(hindi) that implies optimum tree also has a cycle but that is not possible. Because it is a tree, so that is the contradiction. (hindi) There is no place where these 2 sets differ. Because, you said, if these 2 sets differ, let us look at the first place where they differ. If they differ, then let us look at the first place where they differ and apply this (hindi) that means they do not differ at all, if they do not differ, these to case are the same. So, Kruskal's algorithm finds the best tree. (hindi)

(Refer Slide Time: 47:47)



$$\begin{array}{ccccccccccc}
g_1 & < & g_2 & < & g_3 & \cdots & g_{i-1} & < & g_i & < \cdots & < g_{n-1} \\
\text{|||} & & \text{|||} & & \text{|||} & & \text{|||} & & \vee & & \\
f_1 & < & f_2 & < & f_3 & \cdots & f_{i-1} & < & f_i & < \cdots & < f_{n-1}
\end{array}$$

Case II : $f_i < g_i$

f_i is distinct from $g_1 \ldots g_{i-1}$

Why did Kruskal not pick f_i?

Bcoz $f_i \cup \{g_1 \ldots g_{i-1}\}$ contains a cycle.

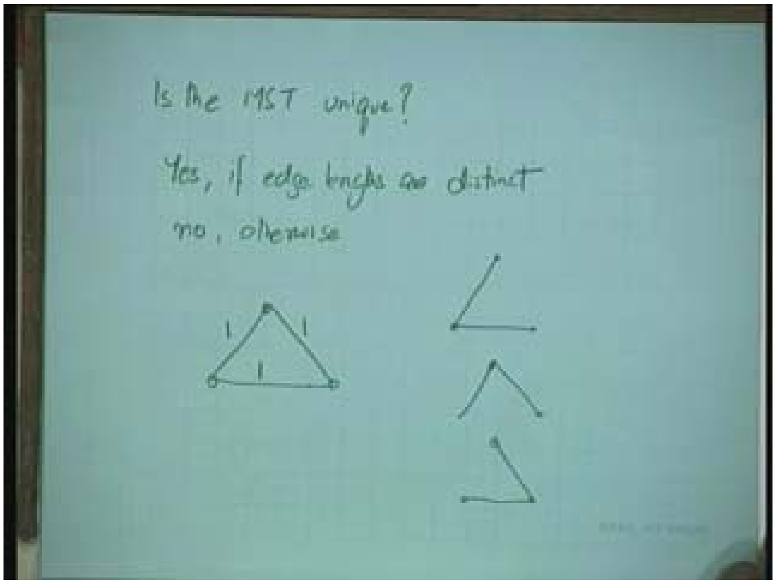
$\equiv f_i \cup \{f_1 \ldots f_{i-1}\}$ " " "

$\Rightarrow$ opt tree also has a cycle

Now, note that there is no notion of the term the best tree. Should I use the term the best tree or a best tree. (hindi) So, is the minimum spanning tree unique? (hindi) Yes, listen to me carefully (hindi) edge lengths are distinct. Is the MST unique? Yes, if edge lengths are distinct. No, otherwise.

Why no, otherwise? (hindi) very simple. So, if edge lengths are not distinct, then you could have many different minimum spanning trees. But, if the edge lengths are unique or distinct, then you have a unique minimum spanning tree. (hindi) That follows from the proof that we have seen, they will be a unique minimum spanning tree. (hindi) Great, (hindi) we have proved the correctness of Kruskal’s algorithm. (hindi)

(Refer Slide Time: 50:12)



Is the MST unique?

Yes, if edge lenghts are distinct

no, otherwise

Now, we have to figure out (hindi) when I include an edge, how do I ensure that no cycle in formed. Basically, that is the question, how do we check if a cycle is formed when an edge e, let say, an edge e, u comma v is included and we have to do this quickly. (hindi)

When will be a, when will a cycle be formed? I am trying to include this edge, u, v. (hindi) When u and v are already connected (hindi) Cycle is formed if and only if u and v are already connected. Yes, already connected, (hindi) u and v are in the same connected component. Connected component or component … (hindi)

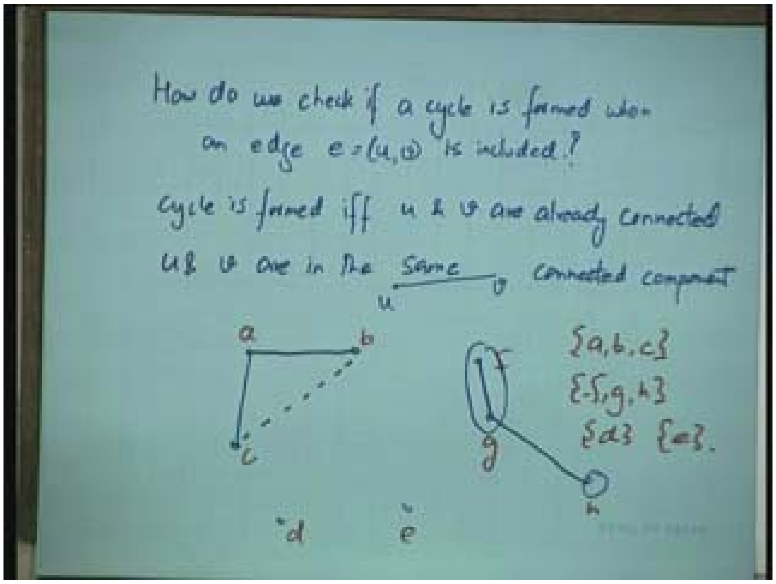
So we are, what we are going to do is, we are going to maintain the collection of components. So note that, so, what is happening? In Kruskal's algorithm, suppose, these were my vertices (hindi) Does not matter. This is or no matter, what 3 edges I pick, it will always have n minus 3 connected component. (hindi) 3 edges which form of a forest, then the number of trees in the forest. See, if I have a forest containing some k edges, then how many trees are there in my forest? n minus k, this you can prove very easily.

Basically, (hindi) I combine, 2 connected components into 1 connected component. That is the way you have to think of, I combine 2 connected components into 1 connected component (hindi) So, the number of connected components reduces by 1 every time you take an edge. (hindi) So, (hindi) at each step (hindi) with every edge we include. So, what we are going to do is, to maintain the connected component we have at this stage, how many connected component have at this stage? 4.

I am going to maintain them as 4 sets of vertices. What do I mean by that? Let me give these vertices names, a, b, c, d, e, f, g, h. So, I am going to have 4 set, abc is 1 set, fgh is another, d is the third and e is the fourth. These are the 4 sets that I have at this point.

I am just interested in the sets of vertices. The sets of vertices in 1 connected component. They form 1 set, as you can see, these sets are all disjoint and there union is the universe, they form a partition of the vertices sets. Now, what is it that happens at each step? When I try to include an edge (hindi) what do I have to check? (hindi) So, this is the data structure we have to maintain.

(Refer Slide Time: 57:39)



How do we check if a cycle is formed when an edge $e = (u, v)$ is included?

cycle is formed iff u & v are already connected

u & v are in the same connected component

$\{a, b, c\}$

$\{f, g, h\}$

$\{d\}$ $\{e\}$.

We are going to discuss it in more detail in next class. You just keep this in mind and we will see how do to this data structure and what time does it give us for Kruskal’s algorithm. So, with that we are going to end today’s discussion. We are going to continue the discussion on this data structure in next class.