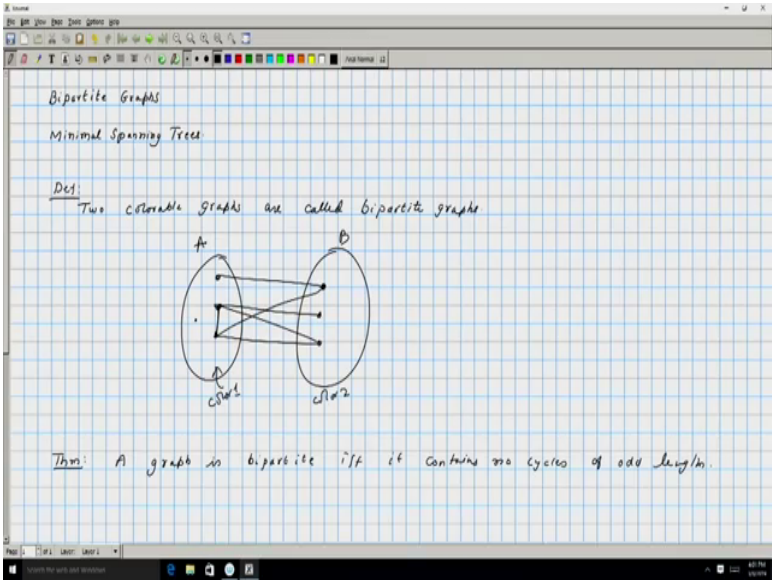
Discrete Mathematics
Professor. Sajith Gopalan
Professor. Benny George
Department of Computer Science and Engineering
Indian Institute of Technology Guwahati.
Lecture 13
Bipartitie Graphs

(Refer Slide Time: 00:32)



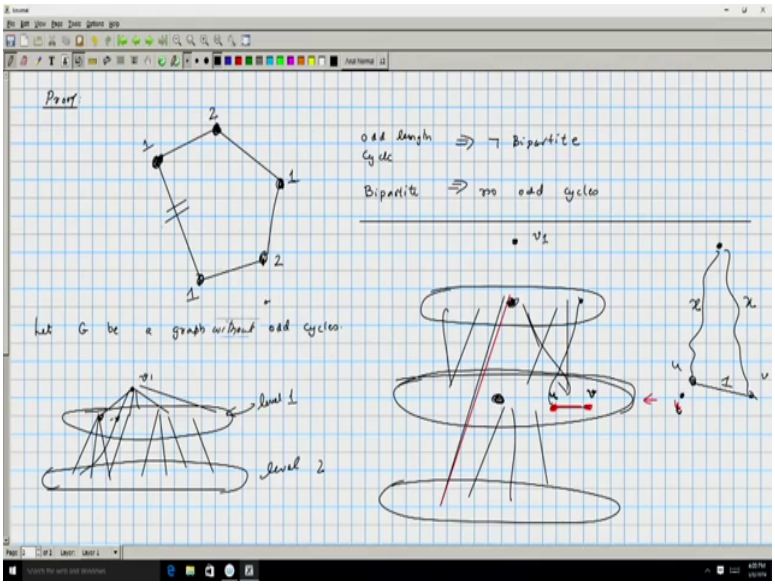
So in this lecture we will learn about two things, one is about bipartite graphs the second thing as about minimal spanning trees. These are the concepts that we are going to learn in this lecture. Let us first define what are bipartite graph? In the previous lecture, we talked about coloring or coloring of vertices of a graph. Graphs, which can be colored using two colors, which can be vertex color using two colors are referred to as bipartite Graphs. So, lets us write down the definition two colorable graphs are called bipartite graphs.

So, if you have any graph and it can be colored using two colors these are called as bipartite graphs. And, the reason why they are called bipartite graph is because, it naturally splits into two parts. Let us say these are all vertices getting one color so, let us called as color 1, and these are all vertices getting color 2. Now, if you look at the edges of graph every edge is of this kind one end is in this particular set let us call this is A this is B, and, the other edge will be in B. This is how every bipartite graph would look like.

One edge in the one part and other edge in the other part, because, if you had two vertices in the same part, then of course these where the colors given to graph, and, if you had an edge in

which both vertices where are on one side, then that would essentially mean that edge is not properly colored, its ends point received the same color, so that is not allowed. So bipartite graphs are essentially two colorable graphs. We will give alternate characterization to bipartite graphs. So, we write it as a theorem and we will prove it, a graph is bipartite if an only if it contains no cycles of odd length, okay? And, look at all the cycles of graph if none of those cycles are of odd length, then it will invariably be a bipartite graph, okay. And, if you take a bipartite graph that cannot contain any cycles of odd length, let us try to prove this here.

(Refer Slide Time: 03:50)



One direction as E C, suppose let us just imagine that there is an odd cycle. Suppose this is one particular cycle in the graphs cycle of odd having odd number of edges. Now, suppose this graph was colorable, then all these vertices would have got some color. So, if you take this vertex one of these vertex start at any vertex, if, you give color 1 to this its neighbors must surely get a different color. So, this must be of color 2 there is no other option.

And, the next vertex should be of color 1, and the next should be 2, and the next should again be 1. And, that would cause this particular edge to be colored in such a way, that both its vertices have the same color, so, that is not a valid coloring. So, this would happen in any odd length cycle. If, you have a cycle of odd length you start with any vertex, by the time you reach back to the vertex, you will see as the colors have to alternate the last edge cannot be colored properly, okay?

So, odd lengths cycles would imply that the graph cannot be be colored using two colors, and, hence the graph cannot be bipartite. So, we know that the odd length cycle implies not

bipartite. So, the counter positive of the statement would mean bipartite implies no odd cycles. Now we need to show the other direction that is, let us take graph, which does not contain any odd cycle that will be bipartite, okay? So let us look at any graph and we are going to color the edges in a particular order, okay?

So, we take any particular vertex, let us call this is v1. So, we are now looking at a graph without any odd cycles and we will show that it is bipartite, okay? So let us take any particular vertex of such a graph so, let G be graph without odd cycles and let us choose a particular vertices v1. We are going to construct an ordering and for this we will first look at all the vertices of vertices which are adjust into v1. That is what we will call as level 1. Level 1 consists of all vertices which are neighbors of v1.

And then we will look at the neighbors of vertices in level 1 that will be a level 2. Look at all their neighbors except of course the neighbor v1, we will look at all the other neighbors and that we will be putting in the next level, and we continue this so we will get something like a level arrangement of the vertices. This is the level 0 that is vertices v1 and there are level 1 vertices its neighbors will give you the level 2 and so on. So, this will have some number of levels.

Note, that if you look at the original graph and look at its all edges, they will pass from level i to level i plus 1 or, level i to level i minus 1. They cannot jump two levels, because if there was some vertex here and, there was let say there was some vertex at level 1 and it is connected to level 3, then, automatically it would have been in level 2 as well because it is a neighbor, so it would have to be in level 2. So, there is no possibility of jumping the levels.

So, all the neighbors were are essentially account for here. Of course the previous neighbors whatever, I mean, for example it will take a node here some of it neighbors are going to be here and, some of the neighbor are going to be in the layer above it. What we have argued is nothing can go from one layer to this, this is forbidden. Now, can there be an edge between two vertices in the same level is this possible? We will argue that this is also not possible because, suppose there was one such edge that is within the level.

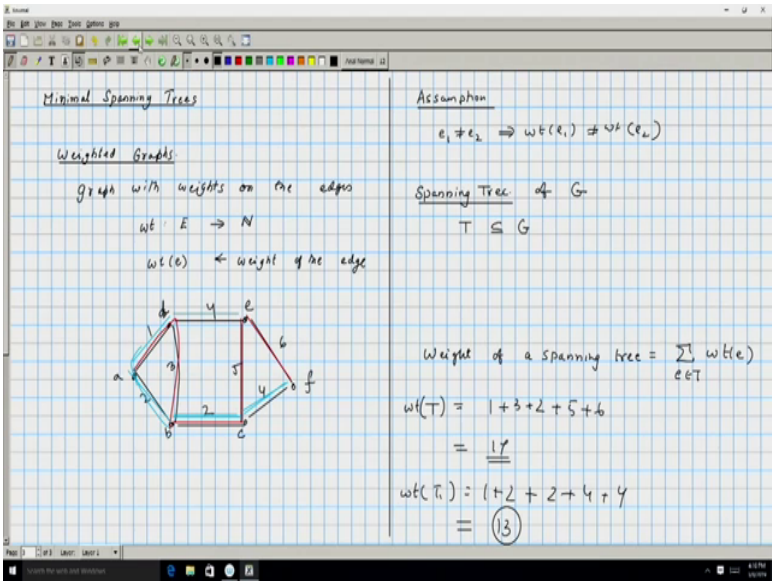
Then these would have some ancestors some common ancestor, of course v1 was there common ancestor, but there could be some common ancestor at a layer closer to this particular level. If you call this as a level i , there is some other level which is closer to i than let say v1 and there is a node there such that that's a common ancestor of both u and v. Now, if you look at the common ancestor and the path to u, there is the path from that common

ancestor to v and the path from common ancestor to u. And both these paths will have length exactly, the height different between these two layers and they will be equal.

And, therefore if you look at so, if this was a common ancestor there is a path common ancestor and there is a path to u, and there is path to v and there is an exhibit in them. Together this forms an odd cycle because this path length is X and, this path length is X and this is of length 1. And, therefore, 2 X plus 1 be an odd number you have a cycle of odd length, but we assume that G is a graph without odd cycles, we wrote it differently but yes this is G is the graph without odd cycles. So, if you take a graph without odd cycles than this is not possible.

So, all the edges will be between adjacent layers and within a layer there are no edges. Now, any graph without odd cycle we can construct this level arrangement. And now if you look at the odd layers and give it one color, and, the even layers and give it another color we can essentially get a valid coloring using just two color, because, between two odd layers the distance is at least two and therefore, there is no edge between them, same implies for between the even layers. So, that is a complete proof that, if you take a graph without odd cycles they can be colored with two colors. It is a two colorable graph or bipartite graph. Now we will see another concept called as minimal spanning trees.

(Refer Slide Time: 11:34)



We learned about trees and trees where graphs such that there were no cycles in it and that they were connected. Now, here when we are talking about minimal spanning tree, we have a specialized kind of graph in under consideration. So, what we will look at is what are called as weighted graphs. So, weighted graph is nothing but a graph with edges having a certain

weight. So a weighted graph is the graph with weights on the edges. That means we have a function let us say w t, which goes from the edge set to natural numbers.

So, w t e would be the, this will be called as the weight of the edge. For example, here is a weighted graph with 6 vertices and weights being as indicated by the numbers above the edges. So, here if you look at the edge a, b and edge b, c they are having a same weight. So, what we will assume in our study of minimal spanning trees is that, the edge weights are distinct, say if e1 is not equal to e2 then weight of e1 is not equal to the weight of e2. So, we can think about our weighted graph as graphs having unique edge weights.

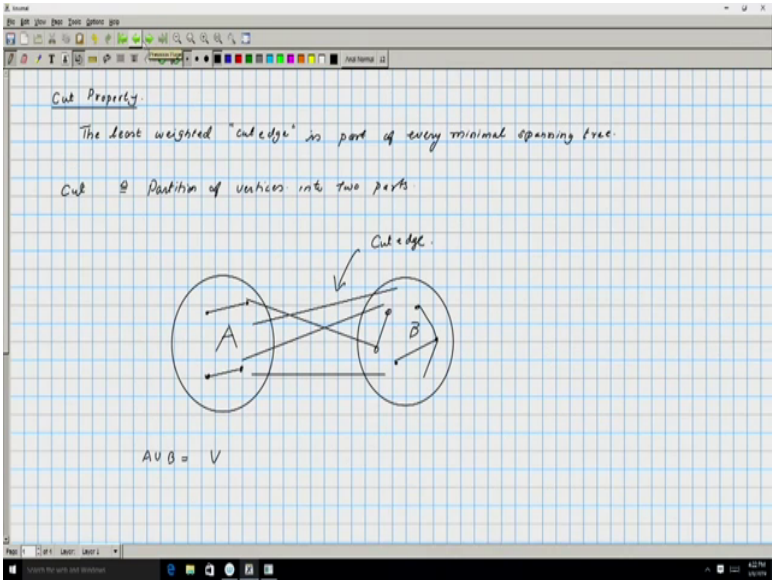
Each edge weight is a distinct number and we will assume that it is a natural number. Now, let us define what are minimal spanning trees? A spanning tree is a tree so t is a spanning tree of a graph G is a tree T which is a sub graph of G. So, there are no isolated vertices they are no isolated components. So, if you take a sub graph of G swings a subset of the edges such that, they form connected graph and it does not have any cycle, in other words you wanted a tree such that every vertex is part of this tree.

For example, if I take the red colored edges that forms a spanning tree, you can see that there are many spanning trees for this particular graphs. Now, the weight of a spanning tree is define as the sum of the weights of the edges. So, weight of a spanning tree is the sum over the edges in T weight of edge. For example, there is red tree that has been described it weight is, so if you call red tree by T, weight of T is equal to 1 plus 3 plus 2 plus 5 plus 6, so that is 17.

Minimal spanning tree is tree which has the smallest weight amongst all the spanning trees. So there could be multiple spanning trees and if we did not assume that the edge weights are distinct, there is a possibility that there could be multiple trees with same weight. And, so all of them will be valid minimal spanning trees. Look at the smallest weighted tree spanning tree that is called as a minimal spanning tree. For example, in this graph if you take another collection of edges.

So, if you take these blue colored edges their weight is going to be if you called them as T1 weight of T1 is going to be 1 plus 2 plus 2 plus 4 plus 4 that’s going to be and 13. So, that is less then T but it is not clear why this should be the minimal spanning tree of this particular graph. Now we will see the certain properties, which will help us, compute the minimal spanning tree of any graph.

(Refer Slide Time: 17:01)



So, there are two crucial properties which will help us identify the minimal spanning tree. The first property is called as the cut property. So cut property essentially states that the least weighted cut edge is part of every minimal spanning tree. So, we need to understand what is cut edges. So, cut is defined as a partition of the vertices. So, when we say partition of vertices we need to get we need to have partition of vertices into two parts, and two none empty parts.

So let us this is your part A and this is part B. So A union B is your vertex collection of vertex, vertex set. And, the edges so, now if you look at the edges of the graph there are 3 kinds of edges, edges which lie completely inside A, edges which lie completely inside B and then there are edges which will go from A to B, these are the only three kinds of edges. And this edges which go from one part of cut to the other part of is called as the cut edge. So, that essentially tell us what is the cut property.

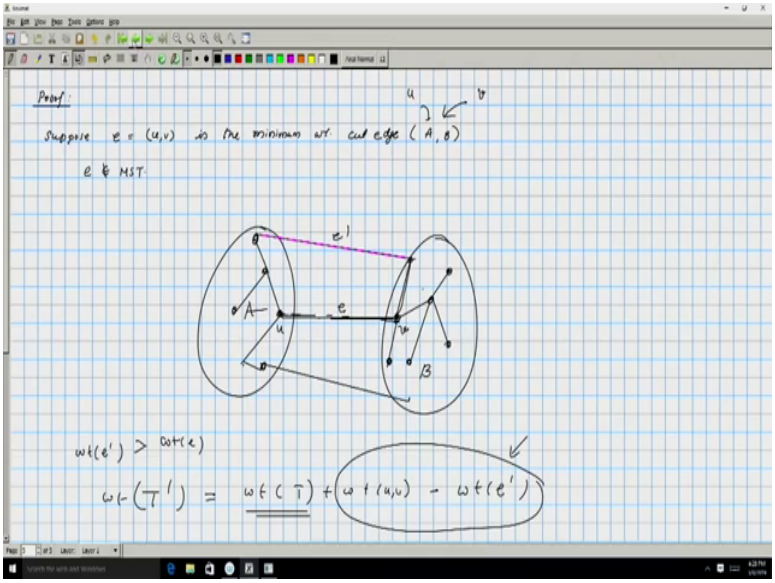
Look at all these edges these cut edges for any possible cut all such edges, put essentially be part of every minimal spanning tree. Here, we are assuming that the edge weights are distinct. And, therefore it will be unique list weighted cut edge. We can always relax these conditions for example, if there are multiple edges carrying the same weight, even then we can apply cut property for that what we need to guarantee is that for the cut that we have chosen, there is a least weighted cut edge, there is one cut edge whose weight is strictly less than every other one, and that cut edge would be present in everything else.

If, you had multiple cut edges minimal once all having the same weight then what we can do at best is that one of those would be part of every minimal spanning tree. So, let us look at

some particular uses of them above lemma. So, if you look at the cut where vertex A is on one side so, if vertex A is on one side and everything else is on the other side and there are two edges which goes from this side to the other side, namely the edge having weight 1 and the edge having weight 2.

The cut property says that 1 must surely be present in all minimal spanning trees, why is this so? We will see shortly. Similarly, if we had taken A and B as a cut as one side of the cut and rest everything on the other side, A B D on one side and C E F on the other side and there are two cut edges the namely the edge 4 and edge 5, sorry the edge having weight 4 and the edge 2 and we can say that the edge B C will surely be part of every minimal spanning tree. Now, let us see how we can prove cut property.

(Refer Slide Time: 22:04)



Now, suppose the cut property is false what does it mean, that means there is some particular edge, which is the least weighted cut edge but, it is still not present in the minimal spanning tree. Suppose, let us say the edge e is equal to u, v is the minimum weighted cut edge across let say the cut we will call as A, B cut means one side is A and the other side is B. And A let us say A contains u and B contains v, that is how u v is the cut edge. And suppose e does not belong to M S T. So, let us draw this M S T.

So some tree is there and then if you add this edge e into it what will happen is, that is going to create a cycle because, spanning tree by virtue of this being a spanning tree there is already a path between u and v because, every vertex was present and this is a tree. Now, if you add the edge u v that is going to create a cycle, can we somehow say that minimal spanning tree

that you currently have is not the optimal spanning tree? So, this diagram was our spanning tree T and we are assuming that u v is minimum weighted cut edge.

Now, minimum weighted cut edge means it is across some particular cut A, B and we want to identify those vertices. So, let us first draw the diagram, so this is our A and this is our B and on top of this diagram we are going to layout our tree. So, tree will have some edges on the A side some edges on the B side and surely because it is a tree there going to be edges going from one side to the other. Now, what we are guaranteed by assumptions is that u v is not one of the edges, but, if add u v to this that is going to create a cycle.

So, let us add this edge u v suppose u is this and v is this, we add u, v to this that is going to create cycle and in this cycle by virtue of this being a cycle there is, I mean, if you just raise the edges there is a path from v to u, u to v is an edge that we added and surely there is path from v to u and that path should surely cause the cut at some point, that means there is an edge which starts in B and ends up in A, by virtue of that being a cut edge for the cut A, B its weight, so let us call this is as e prime weight of e prime is going to be greater than weight of e.

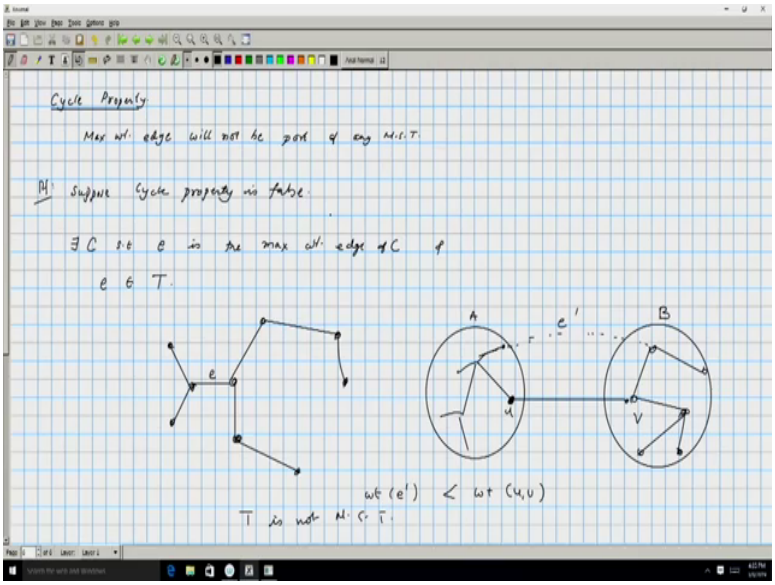
So, now let us imagine the tree which is obtained by removing the edge e prime, so this edge we are throwing out and we are in place is that adding the u, v edge. Now, we will claim that the new object is going to be a tree, it is going to be a tree because this does not contain any cycle. The only cycle that was there was the cycle that we got by adding u, v. Now that cycle has been destroyed by removing the edge e prime so this is a cycle free graph. And, every vertex which is connected earlier still remains connected.

So this is a new tree and the weight of this particular tree. So, let us say the new tree is T prime, weight of T prime is equal to weight of T plus weight of u, v minus weight of e prime, but e prime's weight is going to be greater than the weight of u v, or the edge e and therefore this quantity is a negative quantity. So, weight of T plus negative quantity will give you the weight of T prime. And, this contradict the assumption at T was the tree having least weight.

So it is not a minimal spanning tree, this argument tell us that T cannot be the minimal spanning tree because we have found one particular tree whose weight is strictly less than that of T. That is the proof of cut property. So, construct any possible cut, split the vertices into any possible manner and look at the edges which goes from one side of the cut the other and amongst those the list waited is edge surely is going to be present in all minimal spanning trees that is the cut property.

Later on we will see how we can repeated use this cut property and construct the minimal spanning tree of any graph. Next property, which is again to be going to be a useful property for constructing minimal spanning tree, is the cycle property.

(Refer Slide Time: 28:39)



So consider a cycle in any graph any weighted graph and look at the maximum weighted edge in that cycle, like a guess has to whether that edge will be present or absent in minimal spanning trees. The cycle properties states that max weighted edge will not be part of any minimal spanning tree, so how do we proof this? Let us take a graph G and suppose the cycle properties is false, before we go into the proof let us see a couple of examples.

So, in this particular diagram this graph G, if you look at d, b that is the maximum weighted edge of cycle involving edges with weight 1, 2 and 3. So, the edge marked in green cannot be present, the edge cannot be present because it is maximum waited edge. The same implies to the edge e, c because that is also the maximum weighted edge of the cycle b, c, e, d and e, f is also not part of the minimal spanning tree of part of any minimal spanning tree because that is maximum weighted edge of the cycle c, f, e.

So, in this diagram in this graph if you look at the edges that has been excluded they are fit to be excluded none of those edges can be part of any minimal spanning tree. And therefore if you look at the remaining edges they must be part of every minimal spanning tree, reason being if you remove any one of them, there is no other edges that you can add to make this graph connected. So, this tree indicated by the blue lines are going to be the unique minimal spanning tree.

Now, let us see by the cycle properties. So let us see a proof of cycle property. So, suppose cycle property is false what does it mean? That means there exist a cycle, such that e is the maximum weight edge of that cycle and e belongs to the minimal spanning tree, let us call minimal spanning tree as T, so e belongs to T, e is the edge of the tree T, so this should give us the contradiction. So, let us look at our tree we will draw it out, and some particular edge e which is the maximum weighted edge of some particular cycle is present.

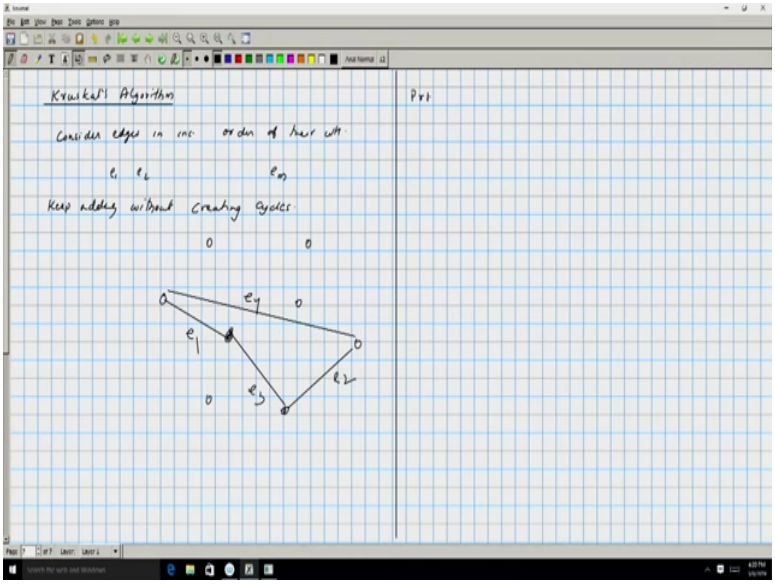
Suppose this was that particular edge, if you remove that particular edge that tree will automatically break into two parts. So that will naturally give us a cut involving or partition of the vertices. So let us say you will draw a tree like this is our edge u, v and there are no other edge going from side a to side b in the tree. So, there are all these other vertices they are all connected to u and all these other vertices which are connected to v so that is our tree.

Our minimal spanning tree was something of this kind and if you remove u, v then that disconnects the graph, but now if you think of the original cycle c of which u, v was a maximum edge, if you look at that cycle and trace out that cycle starting at the edge u v. Then at some point, there should be an edge going back into gong for b to a. The cycle of which the edge e was part of must have an edge, which goes from b to a.

And clearly this not going to be an edge that was the part of tree T because the tree T did not have any edge going from side a to side b other than the u v edge. So, let us call this edge as e prime and since this is part of the cycle weight of e prime is going to be strictly less than weight of u v. Again replacing the edge e prime, replacing the edge e with e prime we get a tree of strictly less weight. So we can conclude that T is not minimal not the minimal spanning tree.

And therefore our assumptions has contradicted. We know that cycle property is also true. So now a (())(34:44) with these two properties cycle property and cut property, we will describe a couple of algorithms to solve the minimal spanning tree problem. Minimal spanning tree problem is you are given some particular graph, some particular weighted graph and you need to compute the minimal spanning tree. These are classic algorithms the first algorithms is called as Kruskal's algorithm.

(Refer Slide Time: 35:03)



We will not bother about the data structures used in these algorithms but instead we will just look at the steps of the algorithm. So Kruskal's algorithm does the following, you look at all the edges sorted by their weights in increasing order. So consider edges in increasing order of their weights e1, e2 and so on. There were m edges then you look at edges e1, e2, em and keep on adding this into the tree. So, initially you have these n vertices which we can think as an isolated vertices.

First vertex first edge that you add is e1 and then maybe you will add e2 and so on. Each time you are adding an edge you will check if there is cycle formed by the edges that are already present. So suppose e1 and e2 and then you add e3 suppose e4 was between these then you are not going to add e4. So, consider the edges in these orders keep adding them without creating cycles and in the end if you had started with a connected graph you will get the minimal spanning tree.

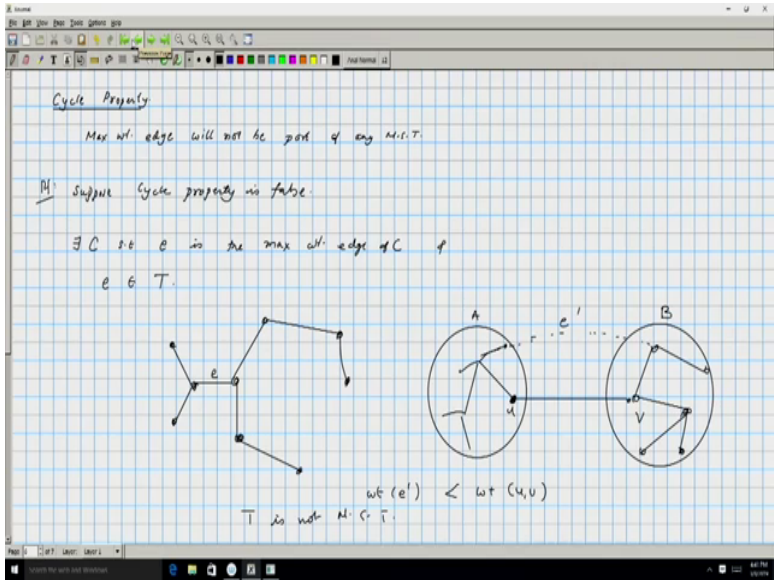
Why is this algorithm correct? The reason is the cycle property. Now every edge that we have removed from this particular tree is an edge which deserve to be removed because that cannot be a part of any minimal spanning tree. This also tells us that if edge weights where distinct that gives rise to unique ordering on the edges and therefore it will created unique minimal spanning tree. So, on graphs where the edge weights are distinct there is precisely one minimal spanning tree and that can found by this method.

Methods correctness rest on the fact that all the edge that we have removed in process to create the tree can, I mean those are the edges which cannot be present in any minimal spanning tree. So if we look at our original graph if, we sorted the edges according to edge

weights the edge weights would have been 1, 2, 2, 3, 4, 4, 5, 6 we need to have some way of identifying which was the weight 2 edge, but if you look at weight 1 and these edges will automatically be added this also can be added.

When you add 3 there is a cycle so that is not going to be added. When you add 4 that is not going create a cycle that gets added. Next 4 is also going to be added and 5 and 6 are going to create cycle, and therefore they will not be going to be added. And therefore these example verify that the tree obtained is indeed is the minimal spanning tree.

(Refer Slide Time: 38:30)



Now Prim's algorithm, is also a very simple way of constructing the minimal spanning tree. You start with the algorithm works in the following way, you start at one particular vertex and look at all it neighbors and amongst the neighbors add the least weighted edge and keep on then let us say if you have added one particular vertex, now you consider this as a set and from this collection find out the least weighted edge and then add that and then consider those three as a set and add the least weighted and so on.

At any stage you have a collection of edges, collection of vertices which are which forms a connected component and from this component look at the least weighted out going edge and that is going to be added to expand this particular tree and in the end you will get the you will get a tree that tree is guaranteed to be the minimal spanning tree. Why it is the minimal spanning tree? Because each edge that you add at any given stage is the minimum weighted edge of a certain cut. So, when you added the first edge if you look at cut consisting of just that one vertex and everything else in the other side you have a valid reason for adding the particular edge that you have added. At any particular stage you have some partial tree and

those partial tree is define a set of vertices and the compliment of that would be the other part. So, if you consider this is a cut a, b the next edges that you are going to add is the minimum weighted edge across the cut a, b and therefore by cut property that is an edge which deserve to be added to every minimal spanning tree.

So, you have learned two algorithms to compute the minimal spanning tree and the correctness of both these algorithm comes from simple properties simple graph properties like cut property and cycle property. We will stop here for today's lecture here, we will continue studying other properties of graphs in the coming lectures.