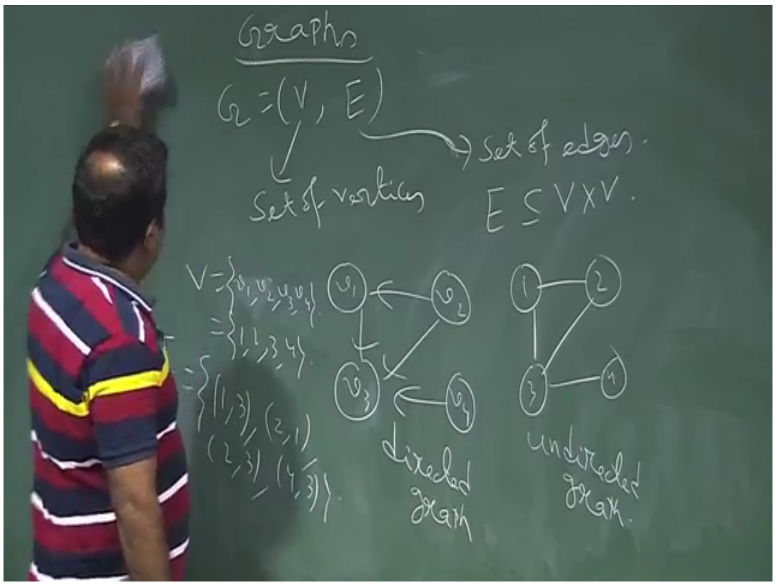
An Introduction to Algorithms

Prof. Sourav Mukhopadhyay
Department of Mathematics
Indian Institute of Technology, Kharagpur

Lecture – 38
Graphs

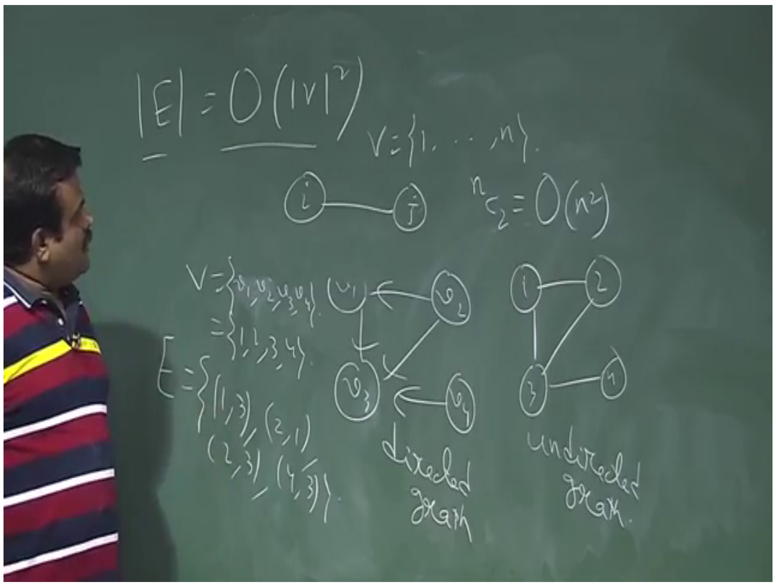
(Refer Slide Time: 00:28)



So, we start we talk about graphs. So, just to recap what is graph. So, basically for graph we have V and E. So, we is the basically set of vertices or it is called nodes also, and E is the set of edges. So, basically there are edges is basically E is a subset of V cross V and there are 2 types of graph directed graph and undirected graph if there is no ordering of the edge then it is called undirected graph and if there is a ordering in the edge this called directed graph.

Like so if we have vertex of v 1, v 2 say v 3, v 4 and if we have like this. So, this is a directed graph or diagraph because you have direction in the edges and the vertex said is V is basically v 1 v 2. So, basically we have v 1 v 2 v 3 v 4 or for simplicity we can just write 1, 2, 3, 4 for simplicity and what are the edges E is said is basically. So, we have a E s form v 2 to v 1, so v 1 to v 3. So, you can say one to 3 like this. So, E s is a subset, subset of Cartesian product of V cross v. So, 1 to 3, then we have E s from 2 to 1, and we have a E s from 2 to 4 and the we have a E s from 4 to 3. So, this is then E s and this is the directed graph. Now for undirected graph we are not having the direction. So, like this. So, suppose this is a this is a graph, so and v 1, v 2, v 3, v 4. So, these are the edges. So, if for undirected graph if 1, 2 is a then 2 1 is a edge like this. So, this is a undirected this is a directed graph, this is a undirected graph.

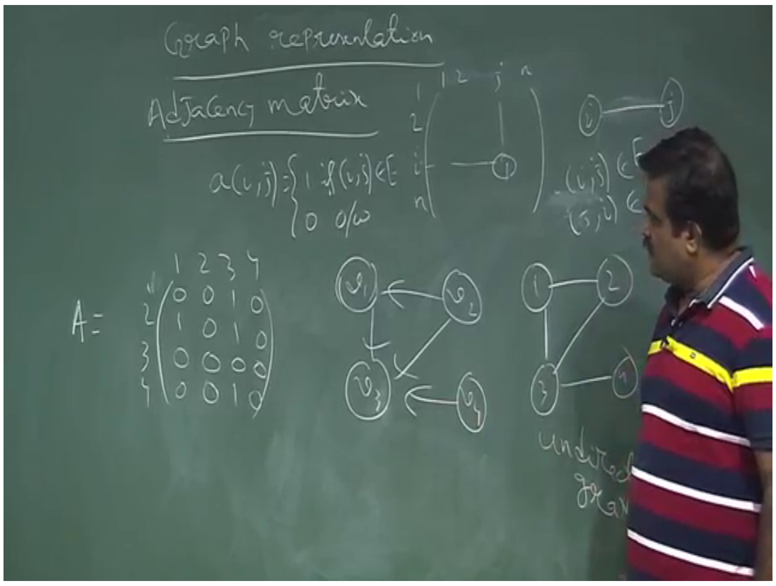
(Refer Slide Time: 03:08)



So, now in a graph, what is the cardinality E, so E is basically the order of V square big o of V square. So, because it can be at most connected I mean complete graph complete graph means every vertex any to vertex there is a edge. So, any to vertex there is edge mean if there are n vertex. So, if we have V say v 1 v 2 V n. So, if it take any 2 vertex i and j if there is a edge. So, then there will be n c 2 edge. So, this is basically order of n square, so that is why order of V is upper bound by order of n square. So, this is for a complete graph this. So, this is the upper bound. So, order of V is bounded above by order of n square.

So, now the question is how we can represent the graph how we can present a graph to a computer or how we can represent a graph. Suppose, we want to give the input of this graph to a computer we want to have a programming we want to have a algorithm where our computed need to take a graph as a input. So, for that we have to so there are two method basically adjacency matrix and adjacency least. So, what is the adjacency matrix representation.

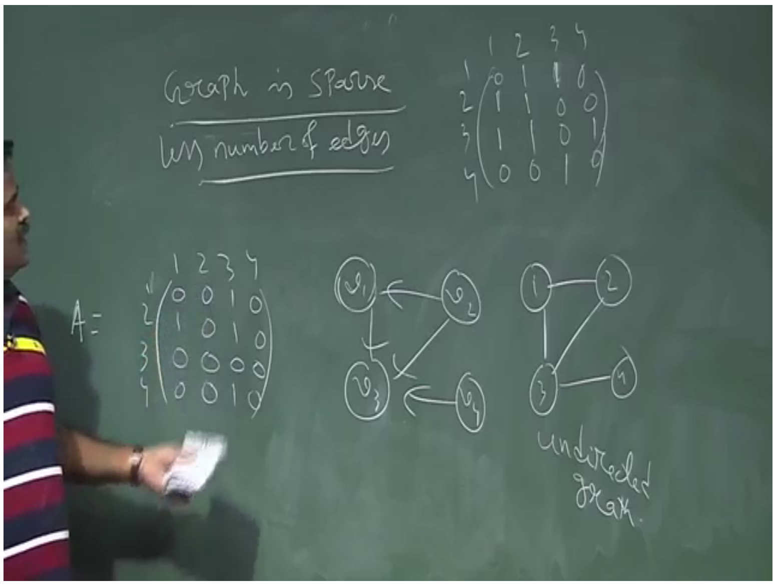
(Refer Slide Time: 04:43)



So, let us to talk about graph representation. So, this adjacency matrix, so adjacency matrix is basically suppose you have m vertices. So, this is a matrix of size m by n. So, this at the vertices so up to n, v 1, v 2 up to n. So, matrix will be 0 on matrix. So, if this is a i th vertex, this is a j vertex. So, i j element will be 1, if i j is an edge. So, this matrix say a i j is 1; if i j is an edge 0 otherwise. So, if you have a edge from i to j, this is the i th vertex this is the j th vertex if there is a h from i to j then that corresponding field will be 1 in the adjacency matrix; otherwise it will be 0, that means there is no edge it will i to j. So, this is if a graph is undirected graph then this matrix will be the symmetric matrix because if undirected graph there is no direction of the s so that means, for undirected graph i j is also an edge, and j i is also an edge.

So, this matrix will be symmetric matrix for an undirected graph. So, for this graph, so what is the adjacency matrix corresponding to this graph. So, there are four vertices. So, the size of the matrix will be 4 by 4 - 1, 2, 3, 4 1, 2, 3, 4. So, this is the adjacency matrix. So, we have from one where is the edge, one there is no edge with one. So, this is 0, there is no edge with 2 – 0. There is only edge from 3 – 0; and from 2 we have edge 2 1, there is no self look self edge, so this is 0. So, this is 1, there is no edge to 4. From 3 there is no edge to any other vertices; all the edges are incoming edges for 3 vertex number 3. For 4 there is now edge to 4; for 4 there is no edge to 1; for 4, there is no edge to yeah there is 3. So, this is the adjacency matrix corresponding to this graph.

(Refer Slide Time: 07:59)

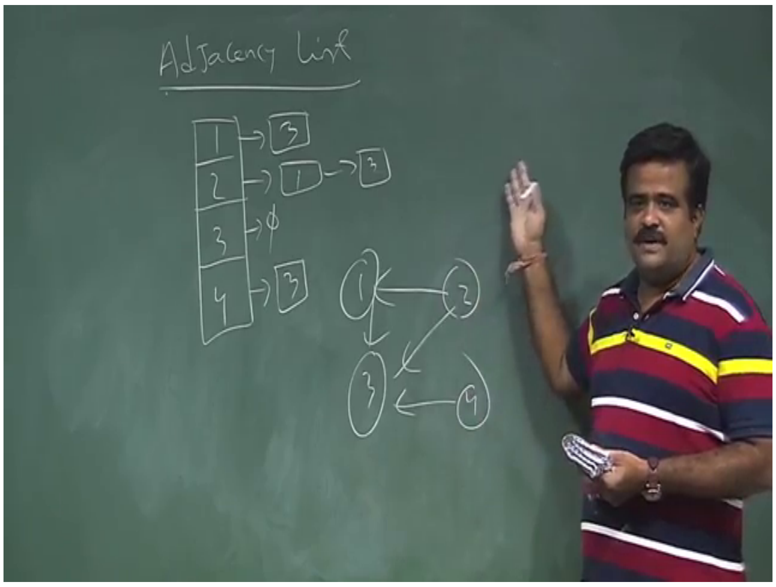


And corresponding to this graph and in the graph is undirected graph. So, corresponding to this graph what it is the matrix 3, 4, 1, 2, 3, 4. So, we have edge between 1 2. So, this is 1 to 1 – 0; 1 to 2 have a edge. So, 1 to 3, we have a edge; 1 to 4 no edge. So, 2 to 1 there is a edge; 2 to 3, there is a edge and 0, 0. And then 3 to 3 there is a edge, 3 to 2 there is a edge, 3 to 3 there is no edge, 3 to 4 there is a edge. So, 4 to 0; 4 to 1, there is no edge; 4 to 2, there is edge; 4 to 3 there is edge; 4 to 4 there is. So, this matrix is a symmetric matrix because for undirected graph if i j if 1 to 2 is edge then 2 1 is also edge, then there is no direction of this. So, this is the adjacency matrix corresponding to this undirected graph.

So, now, the question is whether this is a good way of representing it a graph. So, what is the drawback of this. Now, suppose the our graph is spares graph that means, suppose there is less number of edges suppose the graph is spares graph so that means, less number of number of edges are less, less number of edges. Then what will happen within then this matrix will be a sparse matrix that means, our vertex is more suppose we have 100 vertex. So, this matrix size will be 100 by 100 and suppose we have only 10 edges. So, in this matrix 100 by 100 matrix we have only 10 fields where we have 1, remaining has 0.

So, this will be sparse matrix so that means, we are wasting the memory to have represent base we need to have a array two-dimensional array if you are implementing this in c, we have two different two-dimensional array to have this matrix. So, basically for sparse graph this presentation is not good. So, for that we just have what is called adjacency list that is also one way of representing a graph.

(Refer Slide Time: 10:41)

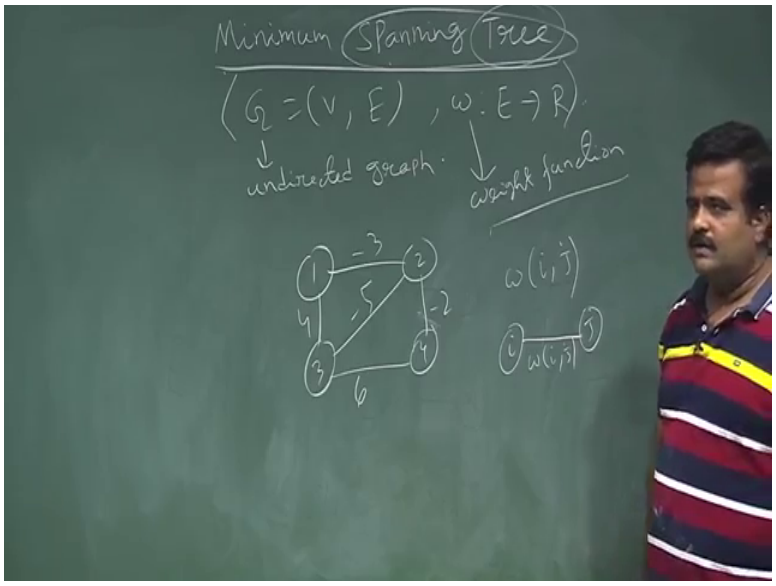


Adjacency list, so adjacency is list is basically so we take a vertex and we have a link list corresponding to each of the vertexes. So, this list contained the vertices which are connected to that vertex. So, for example, if we had that the graph 1, 2, 3, 4, so we have this v 1, v 2, v 3, v 4. So, adjacency list means 1 is connected with 3. So, 2 is connected with 1 and 3. So, and 3 is there is no, this is empty; and from 4 we have a 3. So, this is the adjacency list. So, from each vertex we have a link list kind of thing where that contained the vertices which are connected with this. If it is directed graph, other is if it is un directed graph we have no direction. So, we just take the vertices we are connected to this. So, this is basically the adjacency list representation.

So, this is good if our graph is sparse graph because in that case we have only few vertices so that means, we have the list content few number of edges, a few number of nodes so that way it is good. But if the graph is dense graph if there are more edges than this will be a huge list, then it is good to go for a adjacency matrix, because that is a 0, 1 representation just the bid vector 0 or 1, so that is very nice data structure just a 0, 1 representation.

So, if it is dense graph that means, if the number of vertex number of edges are more then we will go for the adjacency matrix representation; otherwise if it is a sparse graph if the number of edges are list one can go for the adjacency list. Because in adjacency list you have to go for the ling list ling list implementation, but matrix is very easy to handle. So, this is basically 2 OA we can represented graph other by the adjacency matrix or by the adjacency list. Now, we will talk about some graph problems. So, first graph problem, we talk about minimum spanning tree problem.

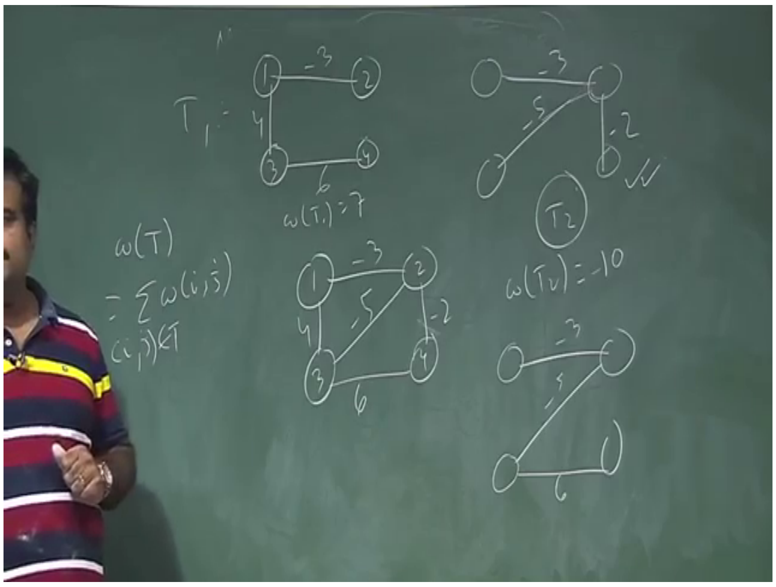
(Refer Slide Time: 13:38)



So, let us talk about minimum spanning tree problem, minimum spanning tree problem. So, for this what are the input, input is a basically we have a undirected graph. We have a graph G, which is basically a undirected graph. And we have a edge OA the on h. So, we have a weight function w which is coming E to R. So, w is a weight function. So, each you have a edge weight. So, now for example, suppose we have a graph like this, this, this suppose we have a graph like this. And we have some weight over we have minus 3 4, minus 5, 6, minus 2 say and these are the vertices v 1, v 2, v 3, v 4. So, at this is the weight function. So, w of i j is basically if i j is an so weight so if i and j is a vertex 2 vertex and if there is a edge, so w of i j is the weight on that edge. So, weight function is also another input.

So, this three are the this is the input of a minimum spanning tree problem. And we have to find out a minimum spanning tree. So, basically you need to find out a spanning tree which of minimum weight. So, what do we mean by spanning tree? So, it is basically a sub graph it is a tree. So, tree means it should not contain any cycle. So, it is a tree basically it is a sub graph it should not contain any cycle and spanning meaning it should cover all the vertices. So, you should get a sub graph which first cover all the vertices. It should cover all the vertices that way it is called spanning tree, and the weight should be minimum among all other. So, what are the spanning tree we can have from this graph. So, if this is the input what are the spanning tree we can have. So, we can have the tree like this we can have. So, tree should not contain any cycle. So, we can have like this.

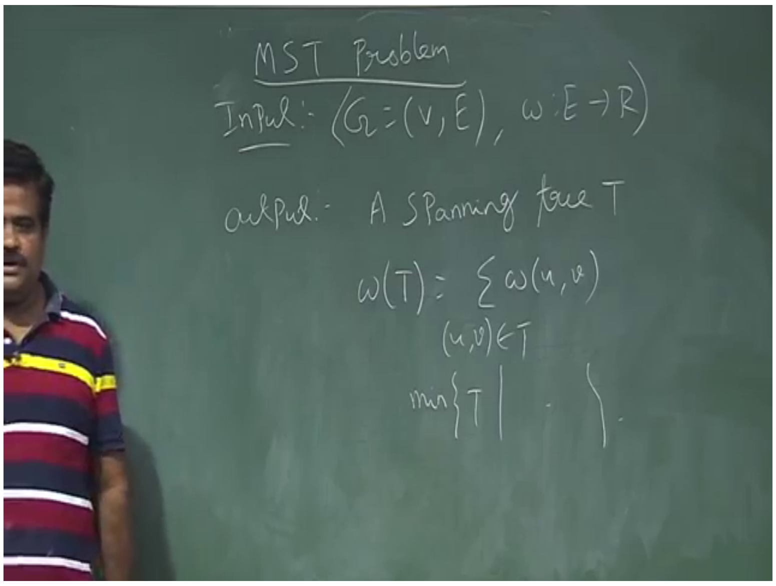
(Refer Slide Time: 16:24).



So, this one, then this one, then this one, this is also covering all the vertices and there is no cycle, so 1, 2 3 4. So, this is the tree T 1. So, this is the weight is minus 3, 4, 6. So, what is the weight of this tree. So, weight of a tree is basically the sum of the weight of i j, if i j is the edge index tree this is the weight of the tree. So, we just basically do the sum of this. So, this is basically 10. So, this is weight of this tree is basically 7, weight of T 1 is 7. So, any spanning tree we have. So, we can have this tree. So, we can this one this one and this one, this is also a spanning tree. So, this is weight is basically minus 3 minus 5 minus 2, so this is T 2, so weight of t 2 is basically so minus 10. And we have another tree also like this, we can have this, then we can have we can have like this, this, this and then we can have this, this. So, this is also a spanning tree, and weight is minus 3 over here minus 5, 6. So, it is not less than this.

So, we can have some other spanning tree also; among these I think this one is the minimum one. So, this is this should be our output of the algorithm, the minimum spanning the minimum spanning tree algorithm. So, this is the problem. So, we have basically we need to get the we need to find the sub graph which is basically tree, not only tree it should be the spanning tree so; that means, it should cover all the vertices and it should give us the minimum edge weight, so that is the problem. So, given a graph given a directed graph we should find out this. So, given a graph, so input is a graph.

(Refer Slide Time: 18:59)



MST Problem

Input:- $(G=(V,E), \omega : E \to R)$

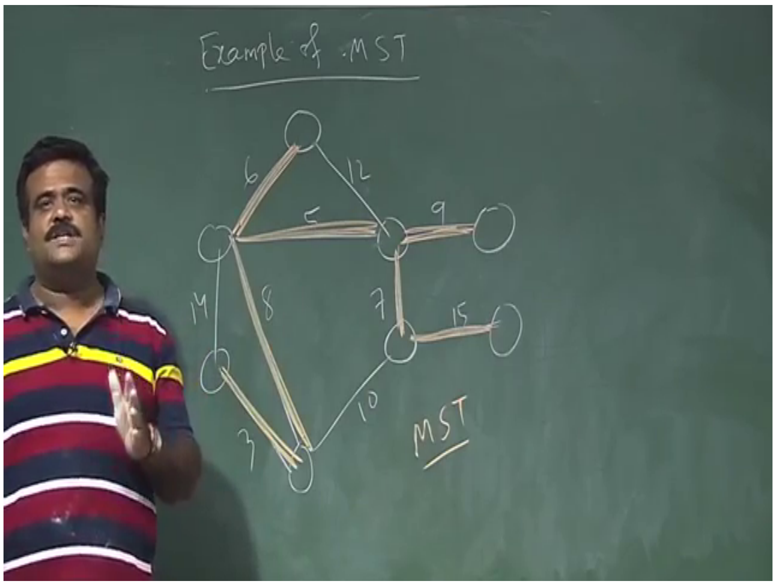
output:- A Spanning tree T

$$\omega(T) = \sum_{(u,v)\in T} \omega(u,v)$$

$$\min\{T \mid \cdot \}.$$

So, this is the MST problem - minimum spanning tree problem. So, input is a graph, G directed graph and we have edge weight. So, this is the input. And the output will be a spanning tree which is a sub graph which is a tree no cycle tree means no cycle and that contained all the vertices such that the weight is minimum w t is the summation of w and this and this is the minimum amount all such t. So, w t is the minimum amount all such t. So, all such we consider all such spanning tree among these switchable give us the minimum weight that is the minimum spanning tree.

(Refer Slide Time: 20:25)



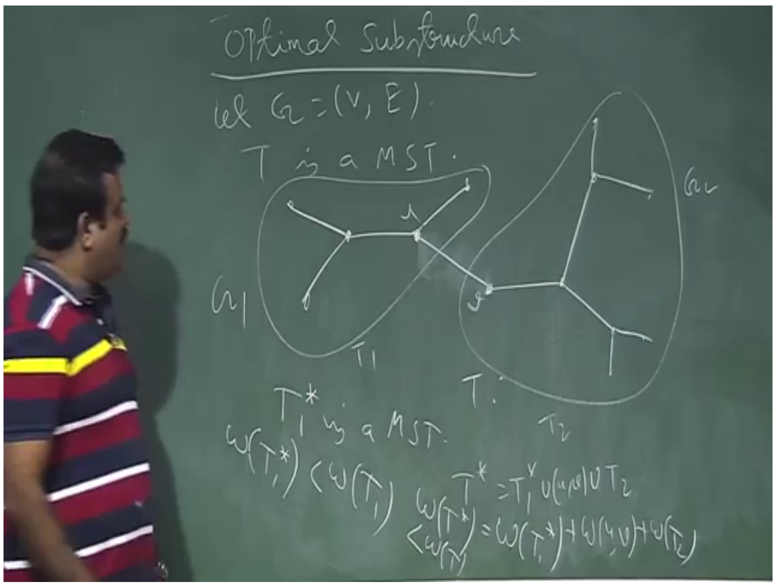
So, let us take an example. So, we have taken a small example let us take a bigger example how we can have a spanning tree of a graph. So, example of a, let us take a bigger graph. So, let us take a little, suppose this is the graph. Suppose, this is the graph with these are the vertices and let us say edge weight 5, 14, 8, 3, 10, 7, 9, 15. Suppose, this is the input and this is the graph with this many vertices and these are the edges and this is the w the weight function on the edges. Now, we want to find out the minimum spanning tree.

Now, can you tell me which are edge should be there in the minimum spanning tree? So, which are the edge must be there in the minimum spanning tree. These two edge must be there in the minimum spanning tree because this is the only edge which is connecting these node with the remaining nodes and the minimum this is the spanning tress. So, it we have to cover all the vertices. So, to cover these two vertices there is no other way other then taking this two edges in the minimum spanning tree.

So, after taking this what we do, so now, we try to cover all the vertices. So, now what we do we just, so it covering this two vertices now if we take this one, it is covering this two vertices. Now, we want to cover these two vertices. Now, if we take this edge, it is covering all the this vertex with a cause 12, instead of that if we take this edge and this edge it is covering these two vertices which is a cost this. So, this is the institution. So, you must take this two vertices. So, this is 5. So, after that, so we have to cover this now how to cover this, we take this because this is now after that to cover this we can take this. So, this is MST, this is the MST, this is the minimum spanning tree. So, we just use the intuition to have this. So, this is the MST.

Now, we want to know the algorithm OA, how we can construction such MST. So, for that we want to see whether we can use the dynamic programming technique which we know for this. So, for that we need to check the hallmark for this. So, basically you need to check, there are two hallmark for dynamic programming problem. So, one is the optimal substructure so that means so let us just anyway if we needed you come back to this graph.

(Refer Slide Time: 25:31)



So, optimal substructure whether we have this hallmark structure. So, what doing in the optimal substructure here optimal sub structure is suppose we have a minimum spanning tree let G be a graph and suppose we have a min and T is a MST. So, if you have the t like this say suppose this one and then we have. So, this is u v like this, suppose this is a minimum spanning tree which is covering all the vertices. So, we have a graph which has have been same number of vertices, but in the graph this is still minimum spanning tree and in the graph we have some more edges what they are, but number of vertices are same. Now, for optimal sub structure what we need to do we need to say that if you take solution a sub if you have a solution of the whole problem then it contain the solution of the sub problem. So, what we do we remove this edge.

So, you remove this edge, and then we consider this tree as T 1 and this tree as T 2. Now, we claim the T 1 is the minimum spanning tree of the G 1, G 1 is the said which is induced by T 1 that means, G 1 contain all the edges which are basically connecting with the vertices from T 1. And G 2 is the all the edges which is basically connecting the vertices of T 2. So, basically G is G 1 union G 2. So, we have to prove that T 1 is the minimum spanning tree of G 1 and T 2 is the minimum spanning tree of G 2 then that will sufficient to show that optimal substructure is there.

So, to prove that let us suppose this is not suppose T 1 is not a minimal spanning tree of G 1 so that means, there is another tree which is has been minimal spanning tree of G 1, so that is the T 1 prime is a MST. So that means, weight of t one prime must be less than weight of T 1. So, what we do we take that T 1 prime and we take this edge and we take T 2. So, we just take this new T star which is basically T 1 prime union u v union T 2 and this T star is a spanning tree of the whole graph G.

And the weight of T star is less than T because weight of T star is basically weight of T 1 star plus weight of u v plus weight of T 2 and this is less than T 1. So, basically this is basically weight of T star is less than weight of T which contradict the fact the T is a spanning tree of minimum spanning tree of G so, that means, this T 1 has to be the minimum spanning tree of the induced graph G 1. And similarly we can T 2 is the minimum spanning tree of G 2. So, this is satisfying the first hallmark of the dynamic programming technique. And the second hallmark is optimal substructure with that also we can verify that there are optimal repetition of this. So, that means, one can thing for a going for dynamic programming technique for finding the minimum spanning tree. But in the next class, we will talk about another powerful technique which is called greedy technique, greedy approach greedy technique to the find the minimum spanning tree.

Thank you.