NPTEL
NPTEL ONLINE CERTIFICATION COURSE

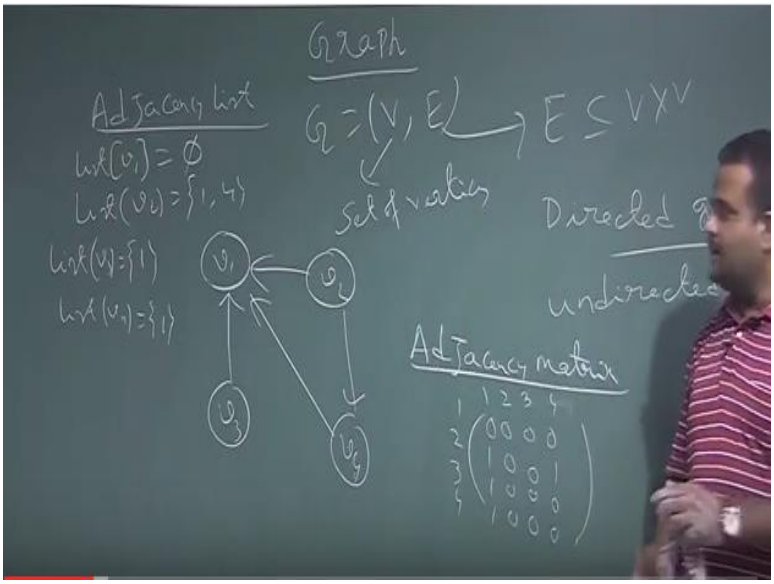
Course Name
Fundamental Algorithms:
Design and Analysis

by
Prof. Sourav Mukhopadhyay
Department of Mathematics
IIT Kharagpur

Lecture 16: Graph Algorithm

Okay, we will talk about graph algorithm today. So let us just quickly recap what is a graph.

(Refer Slide Time: 00:25)



So graph is basically (V, E) so V is the set of all vertices, it is the vertex and E which is basically the edges, it is a subset of V+E. And there are two types of graph, one is directed graph or di-graph and undirected graph.

In the undirected graph there is no order in the edge. Okay, we can take an example of a graph, suppose we have v_1, v_2, v_3, v_4 four vertices 1, 2, 3, 4 so and suppose these are the edges we have, okay. So this is a graph, so we have the edges from 2-1 like this. So this is an directed graph, now if you do not have the direction, then it is called a undirected graph.

So now how we can represent a graph in a computer, so there are two ways we can present that, one is adjacency matrix. So what we do, we basically have a matrix with these are the vertex so there are in vertex and these, so here we have four vertices so 1, 2, 3, 4 and 1, 2, 3, 4. Now whenever we have edge, so from v_1 there is next to any vertex so this is basically 0.

And from v_2 we have edge from v_1 so this is 1, and from v_2 we have edge to v_4 this is one remaining on 0 and from v_3 we have edge to v_1 only remaining as 0, and from v_4 we have edge to v_1. So this is the adjacency matrix representation of a graph. So just whenever we have a 1 that means we have a edge from that vertex to this vertex. Okay, this is problem with this is, if the, this matrix could be a dense matrix, because if there are less number of edge and there are size of the vertex is more then this is a dense matrix.

Unusually we have to store this matrix where many elements are 0, so this two dimensional array. So to avoid this we can just use the adjacency list, we can use adjacency list. So that means we use link this for each of this vertices, so list for v_1, list probably one is empty because no connection from list for v_2 is basically.

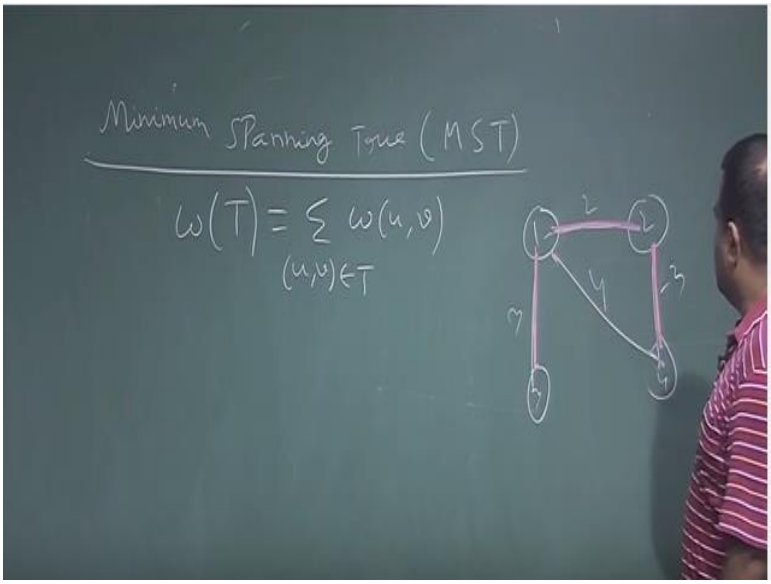
So we have this vertices connected to, we form v_2 where correcting. So from v_2 we have (1, 4), so similarly list of v_3 is basically 1, I mean basically v_1 a list of v_4 is also v_1. So this is the list representation, so our graph is not a fully connected graph or the number of

edges are less then this is good way because this is taking less memory, because we just use the list of this whenever we have edge we have a note for that.

Okay, so now we can see what is the number of edges can be. So if there are, if it is, everybody is having X to if we take any two pair of vertex then if we have a edge, then the number of 8 can be bounded by this okay, this is the complete graph. So log(E) is basically this we will use okay, this result we will use somehow when we talk about iron complexity of some graphs algorithm.

Okay, now we talk about a specific problem, graph problem which is a minimum spanning tree problem (MST) okay.

(Refer Slide Time: 05:10)



So the problem is suppose we have given a graph, this is a directed graph and we have to find the minimum spanning tree, so what is a tree, what is a spanning tree? If we have a graph the spanning tree which is a tree which must cover all the vertices, so that is the spanning tree.

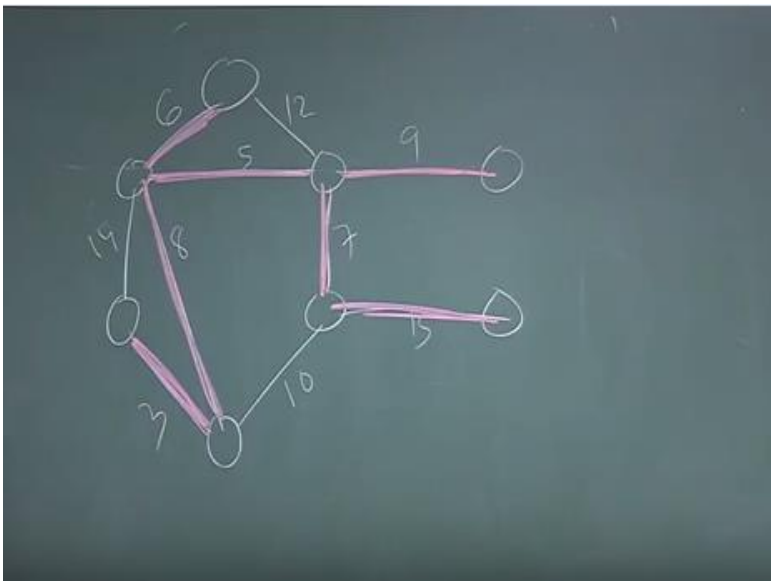
So among all the spanning tree it should be minimum. So w(T) is the minimum among all the, yeah so w(T) is basically, this is the weight of your spanning tree. So if we have a spanning tree and T this is the weight of the spanning tree, so weight, we take the sum of all the edges. Now we are looking for minimum spanning tree, so we may have many trees then among this we want to find whichever is the minimum.

So we can take some examples, suppose this our graph, suppose this is our graph point 2, 3, 5, vertex and if we have say, 1, 2, 3, 4, 5 okay or say 1 is -3 suppose this is the graph and we have to find out the minimum, so what are the spanning tree we can have, so this is, so this one, this one, this one so it must cover all the vertices and there should not be any cycle.

So this is a spanning tree, now this spanning tree weight is, to this weight of this spanning tree is 6+9 okay then we can have this, this, this and that one will be, so we can remove this, so we can have this, this, this spanning tree so this will be the 2, is there any other spanning tree, so we can have the these yeah these two, so among this, this spanning tree is the less weight, so this is the minimum spanning tree available.

Okay so we will talk about the algorithm how to find the minimum spanning tree for a given graph, so before that let us take an example, it is a very small one, let us take a little bigger graph, okay.

(Refer Slide Time: 08:01)



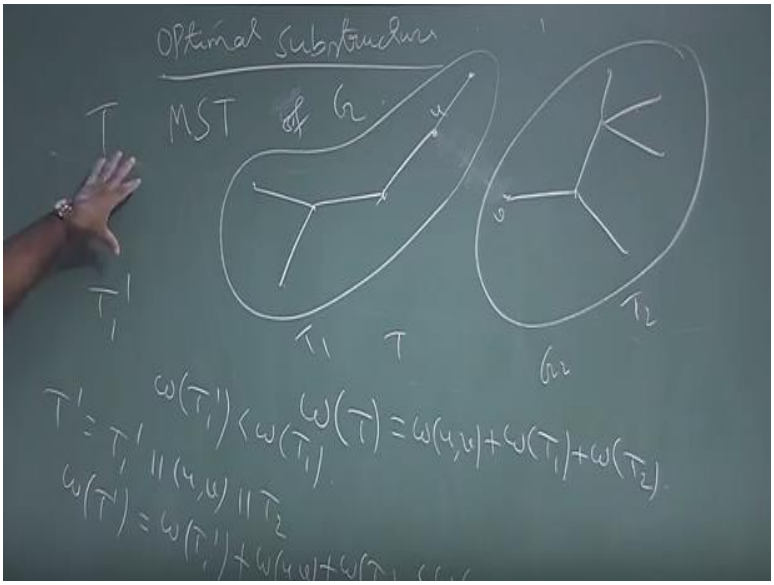
So let us draw a graph with some more vertices okay so this is 6 these are weight A weight 5, 14, 8, 3, 10 okay so suppose this is our given graph and we are going to find the minimum spanning tree of this graph now can you tell me which are the edges must be there in the spanning tree.

These two edges why because this is the only way we can connect the connect these two nodes these two nodes as to be covered in the spanning tree so this is the only way we can connect so these two edge must be there in the minimum spanning tree then what is the next edge we can take this one and then may be this one and then inserted of taking this we can this one and this one may be and then we can take this one okay.

So this is covering all the vertices and this 3 is the minimum weight edge among all the weights, so 3 is there in the minimum spanning tree and there is a theorem that minimum weight should be there in the minimum spanning tree but this, so we will take about greedy approach. So if you are greedy because first attempt the three to be in okay, so that is the greedy choice, okay we will talk about greedy approach screens algorithm, so let us just check before that greedy we want to check whether we can use the any sense of dynamic programming problem can be applied here.

So for that we need to check the what is called, we will come back with this graph, what is called optimal sub structure we check whether optimal sub structure property is there or not.

(Refer Slide Time: 10:36)



Optimal sub structure, this is the first hallmark for the dynamic following problem so suppose we have a minimum spanning tree of a let t, suppose if T is a minimum spanning tree of G and T is like this okay, so this is not capital t, okay among this let us cut one vertex u v okay, I say this is u v cut, now this we will remove.

So this is no more in the tree now this is a this is soft tree, now if we take this sub tree T1 and this is T2 then we take the graph G1 which is covering all the vertices of this is the induced graph by T1 so we take all the vertices over her and there are all the edges which was there in the G but with that these many vertices with the connecting this vertices that is the induced graph of T1 and similarly this is the induced graph of T2 G2 now if we can

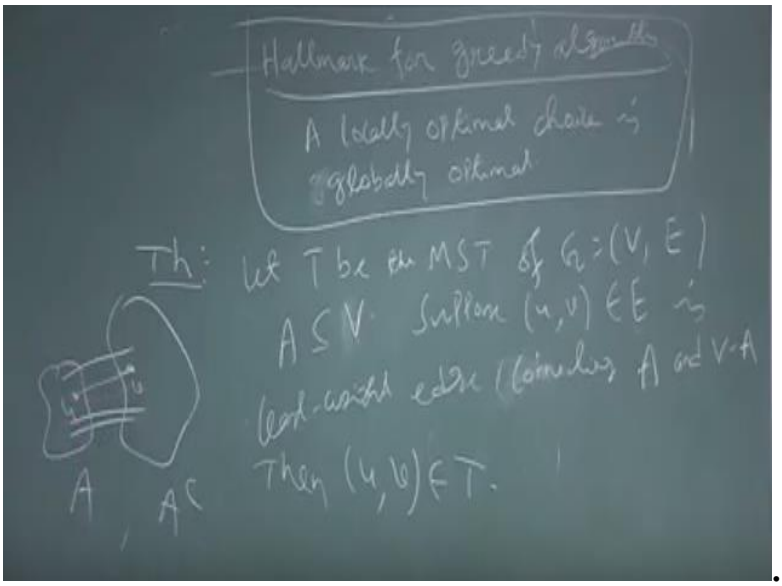
prove this T1 is the MST for G1 and T2 is the MST for G2 then this is the optimal sub structure.

That means if we have a solution for the whole problem and if with that will contain the solution of the sub problems I mean yeah so now how to prove this now to prove this again cut pates techniques so W(T) is basically this weight + W(T1) + W(T2) oaky now suppose we assume that T1 is not a MST of this so that means we have another graph which is T1' which is a MST of this G1 so that means W(T1)' is less than equal to W(T1) now we consider a new path T' which is basically T1 and then we take this u, v and we take T2.

And that weight of this is so this is a spanning tree so weight of the spanning tree is basically weight of this T1 W of this which is less than equal to W(> W(T) so which context the fact that T is spanning tree so this is cut and paste technique so we will prove we have using contention we prove that this T1 is the same spanning two is the base so optimal sub structure is pestering so we could use thing of the dynamic programming so fro that second handle hall mark also we have to check over lapping sub problems.

So but we want to use real powerful technique which called greedy technique to solve this problem so we have to see what is the hall mark for greedy approach.

(Refer Slide Time: 14:23)



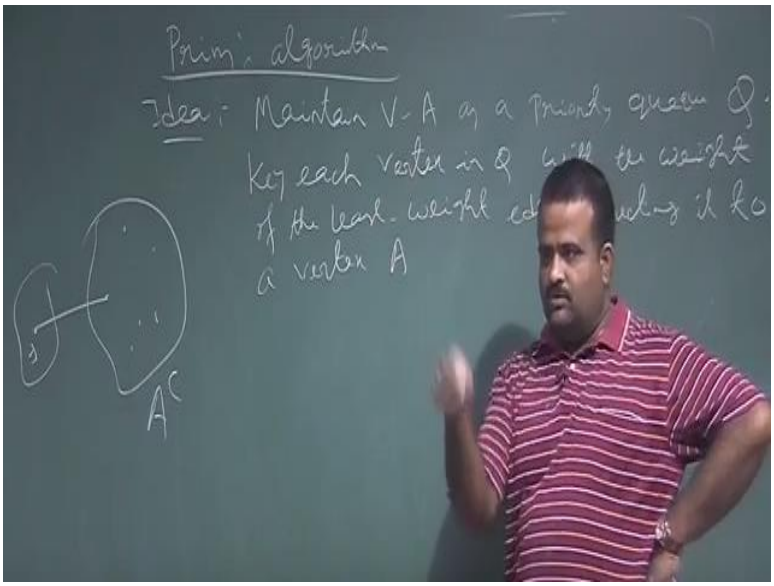
Yeah so hallmark for greedy algorithm so it is telling the locally of a locally optimal choices globally optimal, a locally optimal choice is globally optimal. So if we have this hallmark then we can think of going for a greedy approach so this is coming from so for this minimum spanning problem.

This theorem is telling to go for it, so let TBA, T be the MST of G and you take any subset of V so V is the set of artistes we take and suppose (u, v) is the least to it h connecting a and a complement V-A and then these h must be there in the minimized spanning tree, so that mean we have a set A and this is the A complement the remaining set, so if this is the h which is the bridge h and which is the minimum weight.

That could be another h which are BJ so among this BJ this is the minimum weight age so this theorem is telling this has to be there in the minimum spanning tree so this is the greedy approach so this is the locally optimal choice would be the root globally optimize so we cannot ignore this so we have to add this in the minimum spanning tree, so this is the starting point of the prim's algorithm.

So we will start with the source vertex and we will check the connecting vertex which is the minimum vertex that and slowly we will capture the other nodes and we will grow the tree, okay. So that is the idea, so let us go for the go for the prim's algorithm. So this prove I am not going to details but this proof can be again derived from the [indiscernible][00:17:17] technique.

(Refer Slide Time: 17:21)

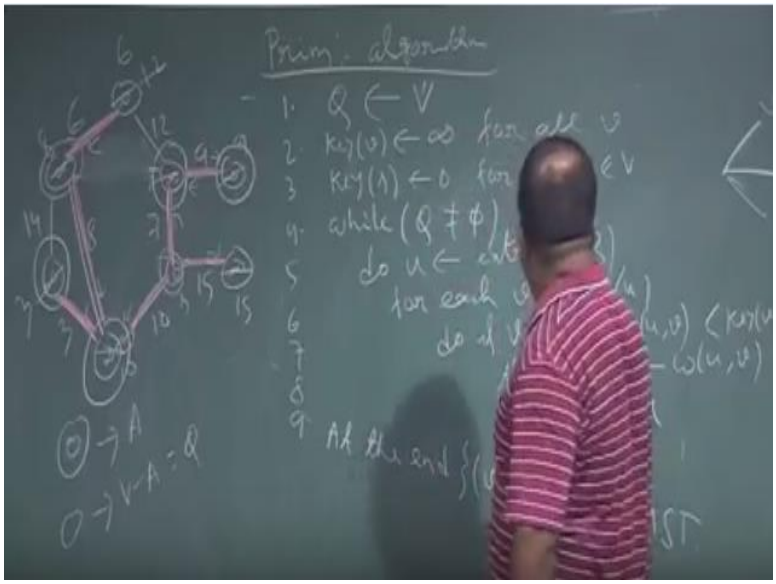


Now I am going to the prim's algorithm which is based on this theorem which is the greedy algorithm, okay. So idea is to so idea is to maintain a priority queue maintain V- 1 elements what is says as a priority queue and we keep each vertices as the so this is the vertices which are not in A, A complement and this is the A, A is starting from S and we key all the vertices which are basically the B or edge minimum weight from this to the any vertices over here.

So that we have to write key each vertex in Q if the weight of the least to it weight h connecting it to a vertex A, okay. So these are so we in initially we put every key values infinity other than this so we put the key value of S is 0 other is infinity because nothing

has explored and then slowly we will capture the minimum weight and we will store we will capture the vertex into this set A and slowly we will grow the tree like this, okay. So let us write the pseudo code for this, okay.

(Refer Slide Time: 19:42)



So this is the, so Q, so initialize which we have, this is the priority Q, we are putting everything there and now we are assigning key value of each is infinity for all V except the key vertex which is the starting vertex for some. We have to start with some vertex and we have to slowly grow the tree and now this is a while loop while Q is not empty.

We extract the minimum and extracting from the Q means we are deleting from the Q and we are putting into the A. Now once it is in the A now we check the other so it is in the A now we check the all the adjacent in your so this U and we check their, this key value whether these could be change or not, okay. Now for each fee which is in the adjacent list of Q, so that means there are some articles G, V1, V2. So these are direct so we check the key value.

So do if, if v is in Q still in Q which is not in A then and if this h is less than p(a) then what we do we, we just replace the key value, then p(v) will be replaced by w(u,v) this is the, this operation is called decrease key operation so we have to go to that position and we have to change this value decrease key operation.

And here so we are changing our key value of v now we have to put a node who is responsible for this change, whose node u, so that we needed to get that h, because it may again change for some other vertices other u so we have to take care of that part in this responsibility vector π(e) is u so this is called response, this is sort of v, v is responsible for this change so that we can get the that h. so finally v at the end this v and the responsibility this will form the MST, okay so this is the code, this is basically we are just starting with the vertices s and we are we can, we are extracting the minimum and each time we are getting one new vertices in here and slowly we are going the tree.

Now let us take an example, the same on the same graph, okay so this was our graph let me write this 14,8,3,10,7,9,15 so this is the graph we want to get the, we want to apply the prim algorithm on this graph. So let us start with this s, suppose this is our source vertices then what else could we taking as a source vertices. Now we put every key value as infinity other than this, this is the initialization step, and then what we do, we just show this is everything so we extract the mean, extract the mean from the Q so this will be s so we put this which are in A and this is in V-A which is basically the Q.

Okay, so now we so this is, so now, so this is, this is has been in A now we check this all the adjacent instead of this of u, so these are the vertices which are this, so now we change their key value this was infinity, now this is 7, this is 15, and this is 10, so this is the key value you have to change by going to the pivot Q. So now we will take the minimum among this key value so this is the minimum, okay and, and for this change these are the responsibility so this is the responsibility vector for this change and this is the minimum.

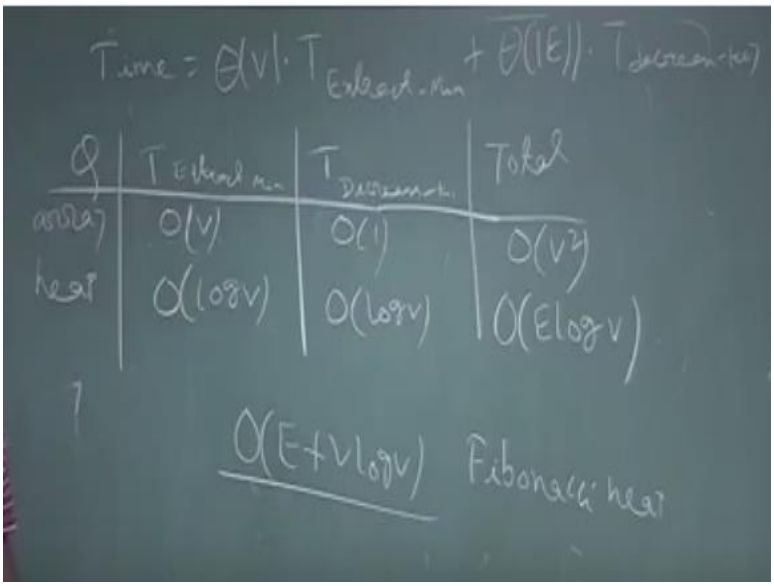
So now we will take this as u now, so this will change this value to 9, 12 and this is already in the Q so no need to do anything and these are the responsibility vector. Now which is the minimum now, 9 is the, so 7,12,15,10 so now we choose yeah, so now we choose 9, no sorry so 9,10 yeah now we choose 9 so and this is the responsibility vector, so this, this is the responsibility vector and now 9 is gone so for 9 there is only one h, so next minimum is 10, so 10 is the minimum so we choose this as u, so for this u these are the, this is already there, so these are the h, so this will be 3, this will be 8, okay.
And this is responsible for, so for this change this is the responsibility vector, okay. Now the next is 3, now for this 3 we have this, this is already there, this is, this is 8 and now this is basically, this is 8, 8 was the minimum so this is 14 so we will not change this and now the next minimum is 8, so for this 8 so sorry we have another vertices over here 5 mean is that. Oh my god, so if mean is that then anyway thing would change here, so yeah, when is that vertices anyway so we just ignore that, so now we have this, this is 8, so 8 this is, this is 12, this is 6 so this will change to 6 and now this is the responsible vector for this, okay.

Now everything is in, now we take all the vertices and their corresponding responsibility vector, so this one because it was earlier change by this, now this is responsible for this, so this why, that is why this will come there this will be there, this and this will again, this will be there this, okay so this is the tree. So we have to check the repeat because this responsibility the vector is changing and for this vector key who is responsible.

So ultimately this is changing many time. So finally the final one we have to take, now what is the time complexity for this code, so the time is basically, so time is basically this is the, this is total this is the initialization phase and this we are doing three times and this is the, so let me just erase this

Okay so this is basically again v times but here we are doing this for, this is we are doing this for degree (u) okay now this degree ∑ of degree (u) is basically order of E, this is coming from LA which is called Hans [indiscernible][00:28:37], so here we are decreasing the key and oh here we are extract the minimum so the time is basically.

(Refer Slide Time: 28:55)



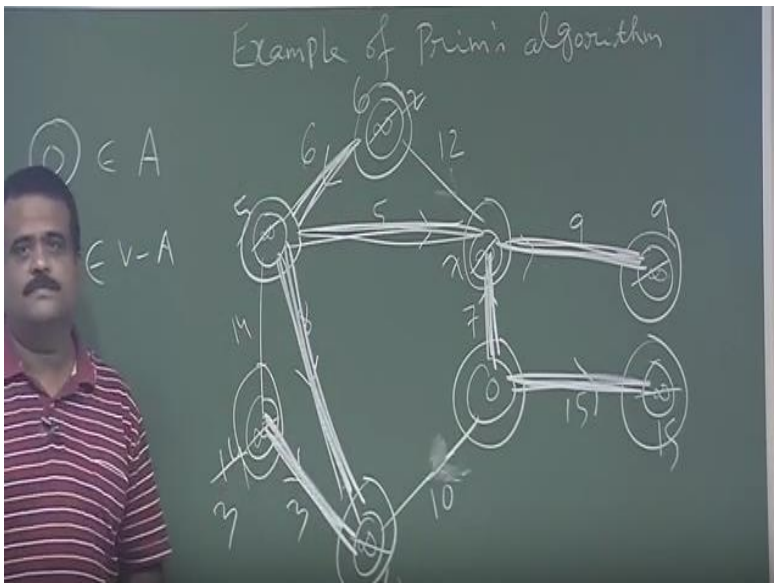
So the time for fields algorithm is basically how much time, time we need for extract the minimum + how much time we need for decrease the key okay so now depending on the, which data structure we are using this halo will be different, suppose we are using a simple array for this pivot q or if you are using a heap, so this will be changing so, so this

Q how we are maintaining this so this is the tree extract the mean and this is the time for dictate the key.

And so this is the total time for Pn algorithm so if we are use the array, array implementation for this Q then this is basically order of v I mean this is constant time because we are just going to that position and we decrease and the total time will be order of.

V^2 and for heap binary heap this is basically order of logv and this is also order or logv we need to call the maximum mean if the mean if and this is order of E logv okay if we use the few and keep and some amount analysis in the washed case it will be E + Elogv is based one but this is using the fibro active data structure if we have probing our syllabus, and this is the washed case for using the data structure so this is the this is the time complexity for the means algorithm.

(Refer Slide Time: 30:57)



Okay so this is we will work out the primis algorithm example, this is the graph we have taken originally so we put every vertex degree as infinity, and except one vertex which is this vertex we are taking, so we are starting this, so this is the vertex which is in A and the remaining vertex in E – A. So we will put the degree of this vertex is 0 and everything we put into the √q, now if we extract the minimum so these will be the minimum.

So this is our now u and this will be in a, so we will start with these and we slowly grow the tree. Now we check the adjacency vertex of this, so these are the adjacency vertex so this degree we have to check, so this degree was infinity now it is 10 and this is the responsible for this so we put a arrow mark here, and this is infinity this is now 15 so this is the responsible vertex for this, this change and this was now this is 7 and this is the responsible vertex for this change of this 7.

Okay, so now who is the minimum, now among these 7 is the minimum so you put this is in A and this is now u so we have to check all the adjacent vertex of u so this is already in A so it will not do anything, so this is becoming 9 and this is becoming 12 and this is becoming 5, okay. So now which the minimum among this so 5 is the minimum so we will put this into the, and these are the responsibility vector so for this changes these are the responsibility vector because these nodes is responsible for.

This to become 5 okay, so now 5 is the minimum now who are the adjacent to this, so this is 12, now we have a 4 so now it was responsible here now we have a this vertex which is responsible to make this as 6, so this is now 6 and then we take this, so this is already in A so we will not be going to do anything so this, so this is now 10 but now it is 6 so it will change into 8, it will change into 8 and now this will be the, now this vertex is responsible for this change, and now.

This is also and this now 14 okay, so these are the adjacent vertexes, we made the change, now the next minimum is 6 so if you take 6 now for 6 all the vertexes, adjacent

vertexes are in A so we are not going to do anything, now the next minimum is 8 so if you take 8 then you see it is basically we have this is already there so this is going to change, this is 3 and for this vertex here this is the responsible vector okay, so now next is 3 so for 3 now we check all the vertexes over here, now this is already, all the vertexes are already there in this list so now we check.

Next minimum, so this is the 9 and this 9 and this is already there is 10, oh 15, so these are the, these are the basically we follow the responsibility vector and this will give us the minimum spanning tree, so this will give us the minimum spanning tree.