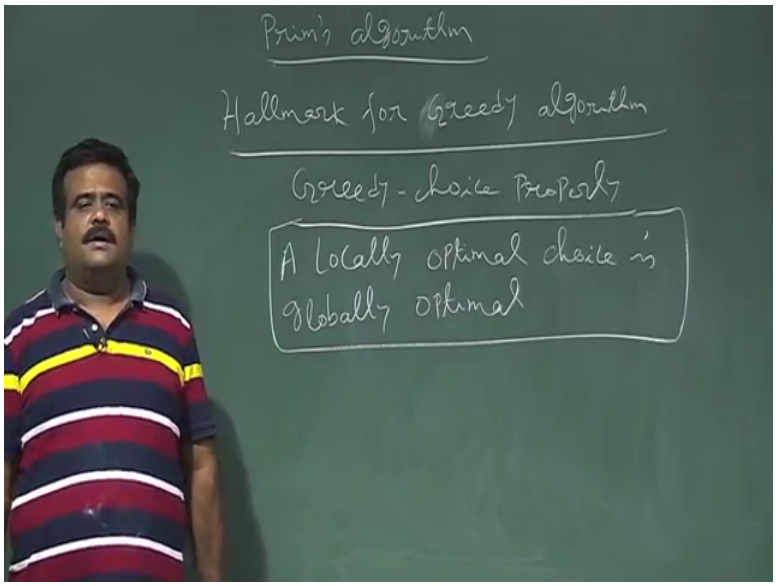
An Introduction to Algorithms
Prof. Sourav Mukhopadhyay
Department of Mathematics
Indian Institute of Technology, Kharagpur

Lecture – 39
Prim's Algorithms

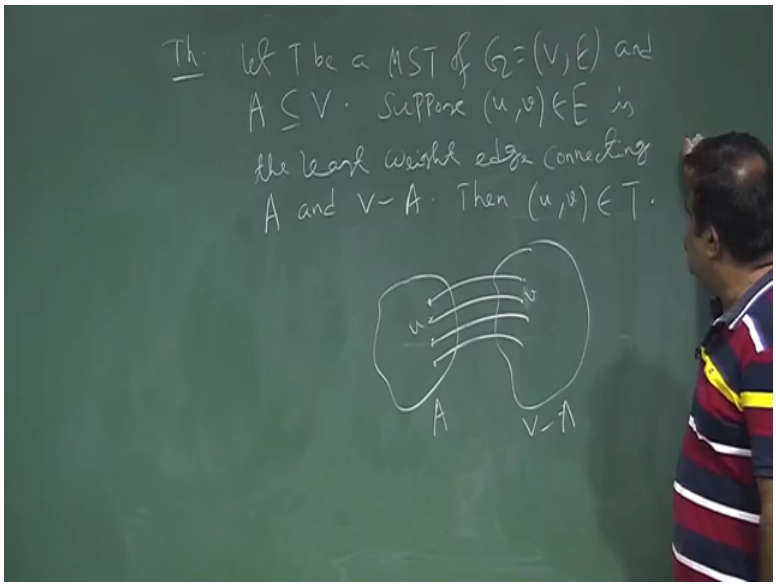
We talk about Prim's algorithm to finding the minimum spanning tree.

(Refer Slide Time: 00:29)



So, this is a greedy approach, so let us talk about Hallmark for Greedy Algorithm. So, it is telling us greedy choice property. So, a locally optimal choice is globally optimal ok. So, this is the hallmark for greedy. So, it is telling us a locally optimal choice greedy choice basically is globally optimal. So, we will just talk about. So, Prim's algorithm is a greedy algorithm; so Prim's algorithm is based on this theorem which is basically greedy choice or which is basically coming from this hallmark.

(Refer Slide Time: 02:03)

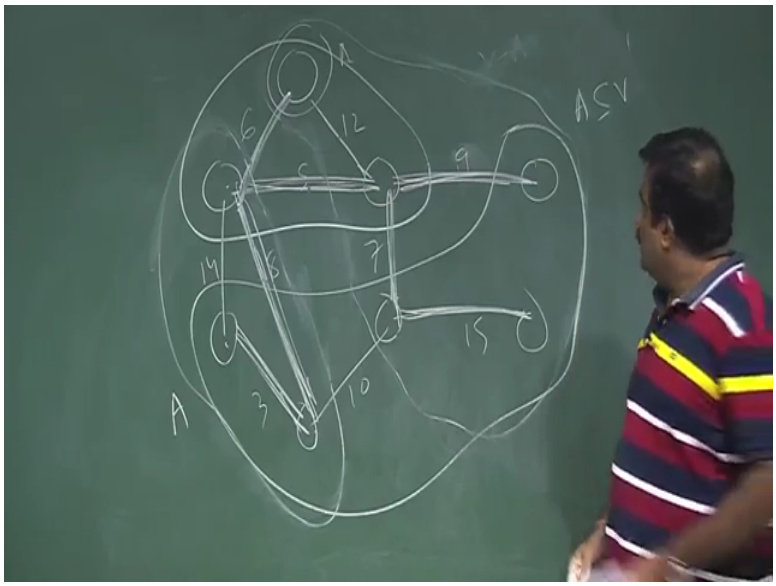


So, this theorem is telling: let T be a MST of a graph G which is basically V comma E and A is a subset of V, any subset of V. And suppose u v belongs to E is the list weight edge connecting V; sorry A and A compliment V minus A. Then this theorem is telling this u v must be in this minimum spanning tree. This theorem is telling this u v must be in this minimum spanning tree. So, what is the meaning of this? Suppose do we have A S edge, we have some vertices and this is the V minus A or A compliment this is the V minus A. So, there are some bridge edges; bridge edge means the edge which is having one vertex in a and another vertex in A compliment. So, there are we can called as bridge edge.

So, among this bridge edge suppose this is the edge u v which is minimum among all this bridge edge, then this theorem is telling this u v must be in the minimum spanning tree. And we have to prove this theorem, but before the let us understand this theorem. So, this is telling us this u v the bridge edge which is the minimum among all the bride edges. And this edge must be in the minimum spanning tree.

So, let us take the earlier example which we have and you see we will prove this theorem.

(Refer Slide Time: 04:31)



But before that and the Prim's algorithm is based on this theorem. So, let us just draw the graph we have in the last class we discuss. So, suppose this is our graph and these are the weight. So, if 5, 12, 14, 8, 3, 10, 7, 15, sorry 7, 9, 15, and we have same. These are our minimum spanning tree; so this one, this one then we at this one.

Now we take some vertices A vertex, like we can take this, this, this vertex as a; suppose this is our A vertex and the remaining is V minus A or A compliment. Now we consider all the bridge edge. So, this is the bridge edge; the bridge edge means vertex one, so one part is in A another part is in A compliment. So, this is bridge edge, this is bridge edge, this is bridge edge. So, among this the theorem is telling the minimum so this is the minimum 5, so 5 means in minimum spanning tree.

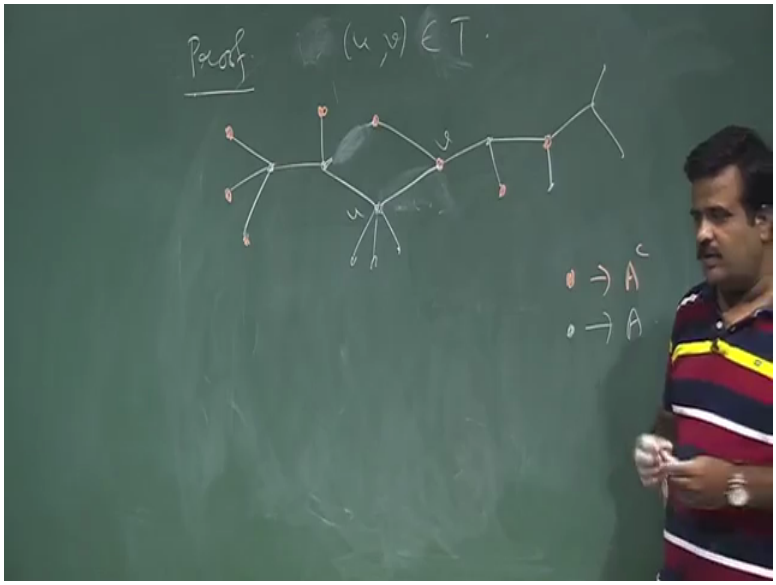
So, if we take another A like this say if we take this is as our A suppose this is our A. Now what are the bridge edge for this? If this is our A, our A compliment is all this, this is our A compliment or V minus A. So, then this is the bridge edge, this is the bridge edge, this is the bridge edge, this is the bridge edge, among this which is minimum 7 is the minimum and we can easily check 7 is in the minimum spanning tree. So, that is the theorem is telling.

So, this theorem is telling if we take a any set A subset of the vertex; so if we take these as A and if we take these as the A say and the remaining are in A compliment then we considered bridge edge this two is the bridge edge so among this, this is the minimum.

So, this has to be in the minimum spanning tree and that is the greedy choice. And that is the locally optimal. So, that is the greedy hallmark. Locally optimal: a locally optimal solution is given us the globally optimal. So, if we start with this vertex we just take this minimum h. So, this is the greedy choice. So, that is the locally optimal choice and this is happened to be a globally optimal. So, that is the hallmark for greedy.

Let us try to put this theorem. Then we will use this theorem to have a algorithm which is called Prim's algorithm. So, this theorem is telling us if we take a set A, any set A ok.

(Refer Slide Time: 08:03)



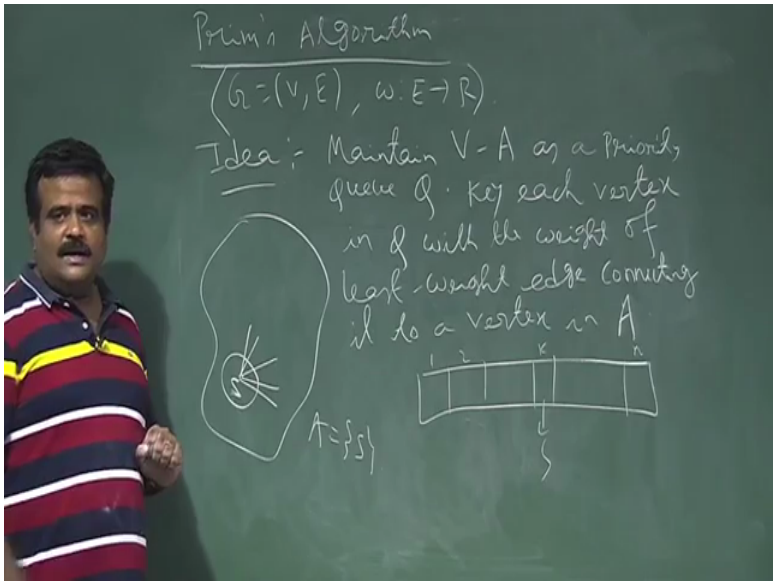
Suppose, so it take set A. So, suppose there are the vertices and it has just draw; so suppose we draw some vertices we are in A compliment. So, these are in A compliment and this one, this one, this one, say this one. Suppose this are in A compliment b minus A. And the white vertices these are in A say. And suppose we have say other; suppose this is the bridge edge this is u and v. So, this is the bridge edge which is minimum weight. And suppose that is not in the minimum spanning tree. Let u v is not in the minimum spanning tree, so we have not having this edge in the minimum spanning tree then we have to list a contradict summation. So, that is the way method of contradiction improve this theorem.

Now suppose this is not in the minimum spanning tree which is basically u v. So, this is the weight of; and the weight of u v is minimum. So, if this is not in the minimum spanning tree and this is a spanning tree. So, basically we have a edge which is a bridge

edge also and that is the edgewhich is connecting from, because we have to cover all the vertices, so we have to cover A and c compliment all the vertices. So, they are has to be edge where that edges a bridge edge and that is we are assuming that is not this one. So, suppose this is this one. So, in set of this if we take this edge and if we remove this edge then this is also a spanning tree and this is the weight is minimum, because this is the minimum bridge edge. So, that contradicts the fact that that weight is minimum, so it is a contradiction. So, in the minimum spanning tree this has to be there.

So, this is the way a contradiction I approve this. Now, based on this theorem we have a algorithm which is called Prim's algorithm which is basically greedy algorithm to find the minimum spanning tree.

(Refer Slide Time: 11:26)



So, let us write the Prim's algorithm. So, what are the inputs? Input is a graph, it is a undirected graph and we have a weight on the edges this is the input and the output will be a minimum spanning tree. So, that is the output. So, what is the idea? So, idea is to maintain a priority queue. So, we maintain V minus A as a priority queue Q and we key each vertex is vertex in queue with the weight of the list weight edge connecting it to to A vertex in a to A vertex Q A, ok.

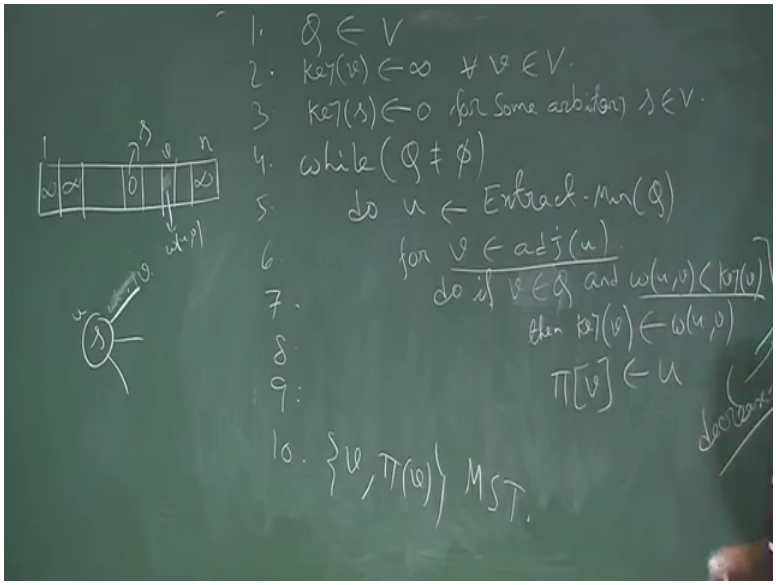
So, connecting it to A vertex in A. So, basically what we do? We basically so we have a graph. So, we basically start with the vertex S and we consider. Now we start with A vertex, so A is constant initially S now. So, that is the starting vertex it could be any

vertices. Now we consider all the bridge edge which is connecting from S to A compliment basically. And that is the key value of the vertices. And we put everything into the Q and the Q is basically maintaining the; this is the priority queue we are it could be array also it depending on how we are implementing. So, we can just have a array, so if V is A vertex . So, you can have a array if there are m vertices, so V 1, V 2, V 3 V n. So, among this vertex we consider A vertex say k V k as S this is our S.

Now we keep the weight of the each vertex as the minimum the at the weight which is connecting from that vertex to A. And now we choose the minimum weight edge and that must be in those minimum spanning tree and that is the greedy choice. So, we cut we capture that vertex in A and slowly we grow the tree. So, we start with vertex S and slowly we grow the tree. So, that is the idea, ok.

Let us write the code then it will be more clear. So, let us write the code for Prim's algorithm pseudo code.

(Refer Slide Time: 15:09)



So, initially everything is a Q. So, this Q is a priority Q, e we can just have a array or the heap implementation for that will come to that. And the key value of V is infinity for all v in V and the exception vertex which is the starting vertex which we put the key value as 0 for some arbitrary S from V. So, we choose A vertex to start, that is the starting vertex. Then, so while Q is not empty; obviously, Q is not empty initially. So, we just

extract the minimum from this Q; extract mean from this Q. And then after extracting means, so then we just update this Q value of the vertex which are adjacent to u.

For each V in the adjacency list of this u; so what we do? Do if degree of v do not degree here we are using the key value do if v is a Q and the u v is less than Q of V then we change then the key of V is basically w u v. And you make it a responsible vector. Now u is responsible for the degree of V change. So, this is sort of responsibility vector. So, this is 6, 7, 8, 9, and finally in 10 at the end this V comma pi V will give us the MST.

So this is the code and this operation is basically this operation is called decrease key. So, if this key value is less we are going to that particular position and we are making the value changing the value. So, this is the decrease key operation. So, this is the pseudo code for prince algorithm. So, basically what we are doing we are starting with the vertex and we are initially we have putting vertex to the key value is infinity, because initially nothing has explore; except a vertex S.

So, if we just implement this using a array. So, we have A vertex this is the array implementation suppose we have n vertices and we chose A vertex S any of this vertex will be S. And initialization is we have this is the key value we have putting everything is infinity except these has 0.

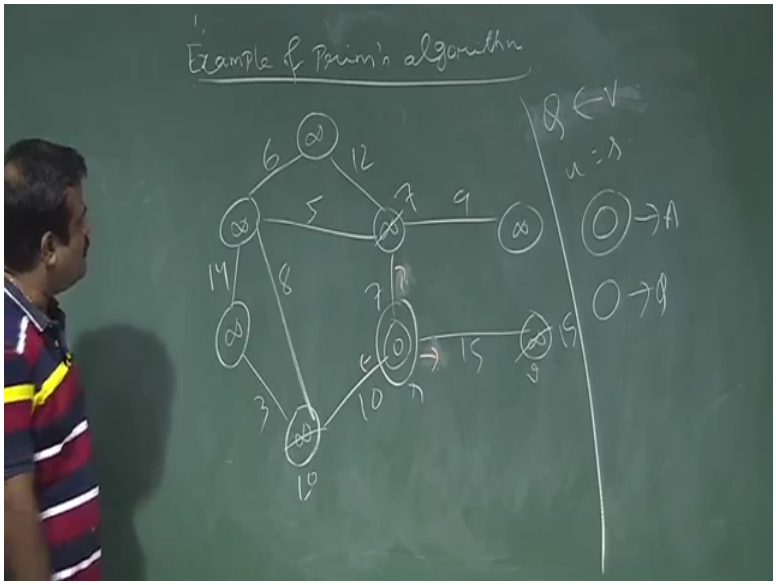
Now, once you extract the minimum since this is 0 so this will be extracted and this we are going to put in basically extract mean means- it is deleted from the Q and it is added in A. So, it is deleted from the Q it is added in A. So, everything is infinity 0 is the minimum, so S will be the first u. Now we consider S is any we chose a; arbitrary vertex as S starting vertex now we consider all the adjacency vertex of S. So, this is our u is the first vertex.

So, now it was having that is key value is infinity initially. Now this has a weight, now if the weight is less than the key value. So, initially it is infinity; obviously, that if that weight is not infinity, so this will be less than so we have to change this key value. So, if this is say this vertex is V then if this infinity now if this has some value, so this value we are going to root over here. So, this way we update all the adjutancy vertex of u. And we make a responsibility vector like for this change, for this degree change this vertex is the responsible. And then we will repeat this until the Q is empty, then we will choose the

next vertex. So, basically we start with A vertex and slowly we grow the tree; that is the idea.

Let us taken example how we execute this. So, this can be implement using the heap also in that case the decrease. So, we will come to the time complexity of this when we analyze the Prim's algorithm. So, let us take a quick example of the Prim's algorithm.

(Refer Slide Time: 20:50)



How it is working? So, suppose we have this graph the same graph let us draw. And these are the weight we have 5, 14, 8, 3, 10, 7, 9, 15. So, this is the input, this is the graph we want to execute the Prim's algorithm. So, we put everything into the priority queue it could be array- simple array. Now, so we put every key value key of V is to be infinity except some starting vertex; suppose this is the starting vertex. So, this is the S starting vertex. And we put everything to be infinity the key value, because nothing has explored this is the initialization.

Now, everything is in Q with the key value infinity except this vertex. Now we extract the minimum, so extracting minimum means. So now, Q is basically all the vertices; now extracting minimum means it is getting the minimum it is deleting from the Q and it is adding in A. So, this vertex is added in A. So now, u is basically s, ok.

So, now we consider all the vertex adjacent to this. So, these are the vertex. So, this is V for this V key value was infinity, now this is 15 so 15 is better than infinity. So, we have

to decrease this key. So, this is now will be 15. And we have to put a mark that this vertex is responsible for this change. So, similarly this will be 7 and this vertex is responsible for this change. So, this mark is needed. So, let us use another color for this mark. So, this is now 10 and these vertexes responsible for this change, ok.

So, now this is our; now this is in A and this is in Q P minus A. Now again we will do the; so Q is not empty, now again we will do the extract mean if you do the extract mean who is the minimum everything is infinity except this is 15, this is 15, this is 7. So, this is our next u. So, we choose this u and this will be extracted from the Q. So, this is our next u, now if this is u now we consider all the vertices adjacent to you. So, now, this is infinity now this will be now 9 and we put a arrow their because for this change this vertices responsible. Now this is another vertex, this is infinity, now this is 12 and we have to pay a arrow for this, and also this is infinity, now this work this is 5 and have to put a arrow for this. And this is already in q so we do not need to do anything, ok.

So, we started with this we captured this vertex. Now which is the minimum? Now 5 is the minimum, so you extract 5 form the Q and put it in A. So, once we extract 5, so we check all the vertices which are adjacent to 5. So, this is vertex, this was infinity, now it is 14. So, 14 is better than infinity so you need to perform the decrease key operation. So, this will be 14 now and this is the responsible vector.

And now this one, this one was 12. Now, we have a better one which is 6. And now these was responsible for this change now. Now for this change, so has to be deleted. Now who is responsible? Now this is the responsible for this. That is why this sign is important, so this responsibility vector. Who is the responsible for? The final change and that will give us the minimum spanning tree.

Now, this is now this one ok, this one is 10 now this will be now 10, this will be now 8, eight is better than 10, now this was earlier responsible now this is now responsible. So, now, this is the situation now who is the next minimum next minimum is basically 6. So, for 6 we have these two vertices adjacency vertices which are already in A. So, we need to do anything in the next minimum next minimum is 8. So, this is our u now this is V is already there we do not need to do anything, now we have to change this because this is 14 now they have a better one so 3 and this was responsible for this now this is responsible for this change.

So now, this is done, now next minimum is 3, but this for this you have added everything now next minimum is 9, next minimum is this. And finally, after this we will follow this length to have the minimum spanning tree. So, this will give us the minimum spanning tree. So, basically the idea is we start with a vertex S and slowly we will grow the tree. So, we start with A vertex S and we slowly capture all the vertices in a greedy way. So, that is the locally optimal choice and it will become a globally optimal; that is a greedy hallmark. So, we start with the vertex S then we e capture this vertex in a greedy choice greedy way. So, slowly we capture all the vertices.

So, this is the example of Prim's algorithm. In the next class we will discuss, we will analyze the Prim's algorithm and the time complexity basically. So, we will discuss the time complexity in the next class.

Thank you.