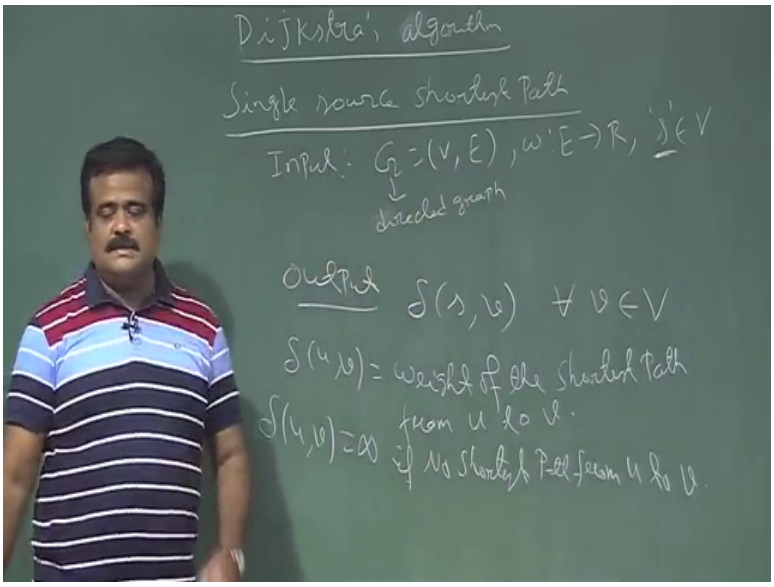
An Introduction to Algorithms
Prof. Sourav Mukhopadhyay
Department of Mathematics
Indian Institute of Technology, Kharagpur

Lecture – 43
Dijktra's

So we talk about one algorithm to find the shortest paths, specially the single source shortest path.

(Refer Slide Time: 00:25)

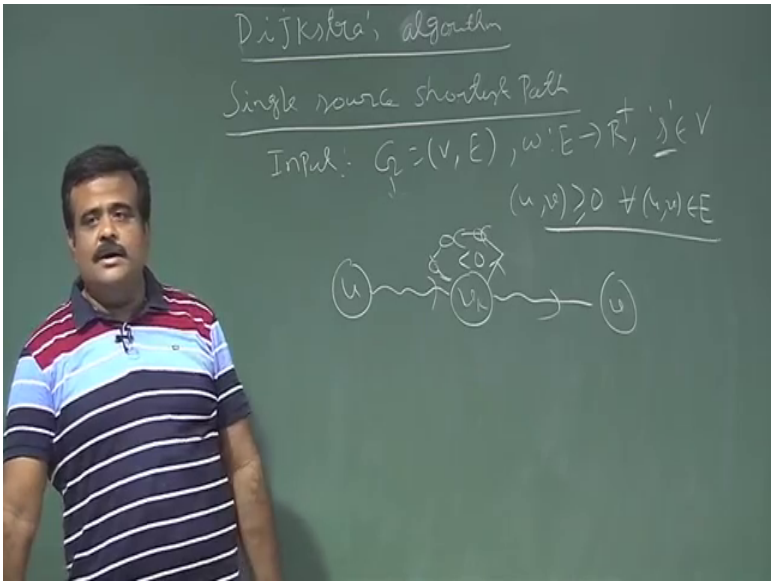


Single source shortest path. So, single source means we have given the source; that means, one vertex is a input also. So, what is the input of this algorithm? Input is basically a graph directed graph V comma E directed graph directed graph v comma E and a weight function E to r and a source vertex. And so, a given vertex is also a input that is the called source vertex.

And the problem is to find the shortest path from that source to all other vertices. So, output will be so, the weight of the shortest path. So, so delta of S comma V for all v. So, this is basically delta E is denoted delta of S comma. So, basically delta of as we know from the last class delta u comma V is we defined the weight of the shortest path weight of the shortest path from u to v. So, u v are any 2 vertices then weight of the shortest path from u to v.

So, first of all for that. So, so if there is no shortest path from u to v we defined this delta V is infinity if no shortest path form path from into V. And in the last class we have seen if there is so, what are the condition there will be no path for no shortest path from u to v if there is no physical path from u to v. So, that is one option. And another option is all though there is a physical path if there is a negative cycle.

(Refer Slide Time: 02:50)

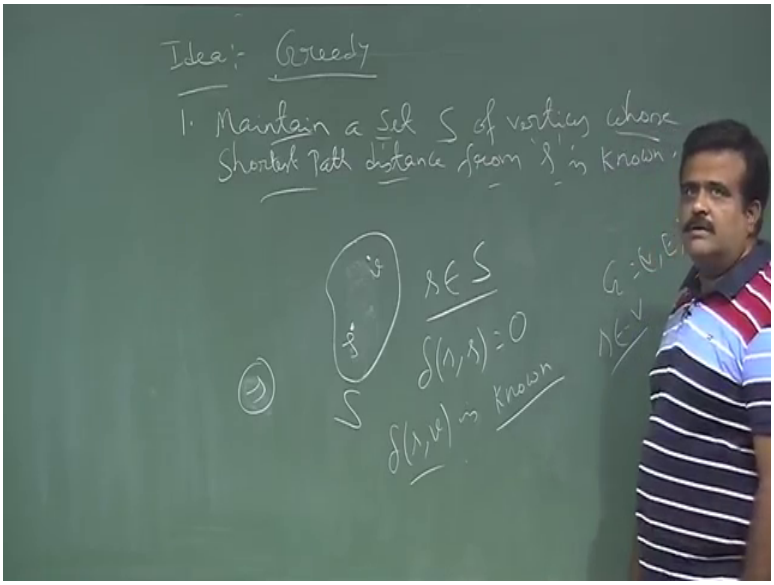


So, if there is a negative cycle between the path from u to v then there will be no shortest path. So, u to v if there is a path, but there is a vertex V k such that there is a there is a negative cycle in the path. If there is a negative cycle then there is no shortest path, because if we can loop in this cycle. If we clime this is my shortest path I will make it another loop. So, they that is why if negative cycle then there is no shortest path exists. So, to have the shortest path. So, we have to be have the physical path and no negative cycle.

So, these algorithm for this algorithm these algorithm cannot handle the negative cycle. So, that is why this algorithm we assume this weight to be negative weight. So, we assume this weight function to be non negative. So, for so, u v are basically non negative weight for all u v belongs to. So, this is one of the assumption for dusters algorithm. For dusters algorithm, because dusters algorithm cannot handle the negative weight cycle. So, that is why so, the input side we take to this weight to be a every weight is positive.

If we if the weight is non negative then there is no question of having the negative cycle in a path ok.

(Refer Slide Time: 04:35)



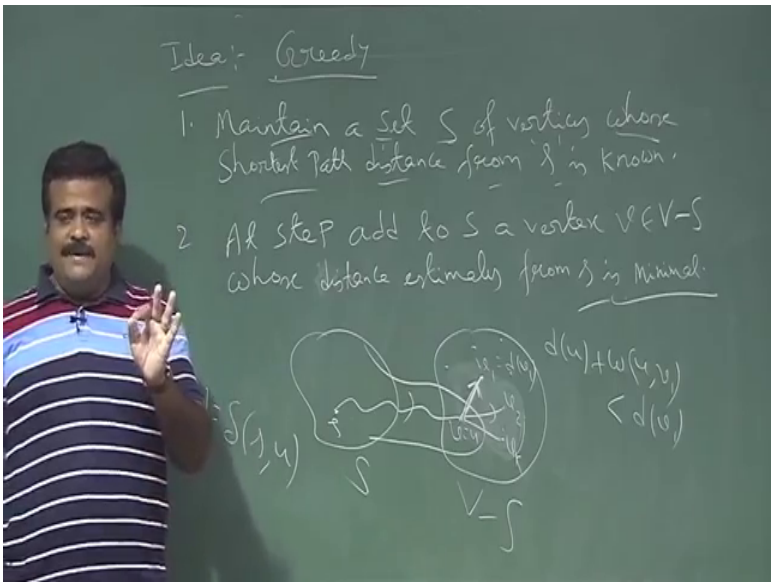
So, this is the input and output is deltas. Now this is a greedy approach. So, this is basically a greedy approach. And the idea is so, idea for dusters algorithm is greedy basically. So, what we do? So, we maintain a (Refer Time: 04:46). We maintain a set S of the vertices whose shortest path, whose shortest path distance from smallest is known. So, so basically you have a graph g and we have all the vertices among this vertices we choose with this is the one of the vertices as a input S. And the capital S basically set of all vertices for which so, for which we know the shortest path from S to that vertex.

Now, now obviously, small S will be in capital S why? Because so, what is the delta of S comma S? Any guess? What will be the delta S comma S? It is 0, why? Because we are not having any negative cycle. There is no negative weight age. So, there is no negative cycle. So, if there is no negative cycle. So, we are at vertex S now what is the best way we what is the shortest path from S to s? So, there is no negative cycle if though there is a cycle that cycle will be positive cycle. So, better to remain as it is. So, delta of S s comma S is 0. So, there is no other way we can reduce the weight less than 0 because there is no negative cycle.

So that means, delta of S comma S 0. So, S will be the first vertex which will be in capital S, because delta of S comma S 0. And then will slowly capture all the vertices

with S. So, that is the in a greedy way. So, let us just, but S is the vertex S is the set of vertices for which if V is in a S then data of S comma V is known. Once we know the delta of S comma V then we capture this V then S. So, we maintain the set S of all vertices whose shortest path distance from S is small. So, that is the our capital S set.

(Refer Slide Time: 07:49)



Then how to grow this capital S? This is basically by the second step at each step. So, we have distance estimate form S to and each step at a vertex to S add to S vertex from V minus S V minus S, whose distance estimate whose distance estimates from S is minimal. This is the greedy choice. So, what we do? So, we have the set S, now this is the V minus S all are vertices. So, small S in S. Now every time we are adding a vertex form. So, this these are the vertices which is having some distance estimate. So, this distance. So, this is a V v is one of the vertex. So, it was having some distance estimate.

Now, we consider all the vertices over here whose this d V, d V we denote by the distance estimate we come to the algorithm the pseudo code d V is denoting the distance estimate from S to this node v. So, we choose one of the vertex whose distance estimate is minimal and that is the free choice. And that vertex we take it into the S from the Q we area maintain a biotech Q from the with the all other vertices other than S and then we are extracting the minimum from this Q. So, this is the idea. So, we choose then once we capture this V in S suppose this V is distance estimate in S. So, so we capture this sorry

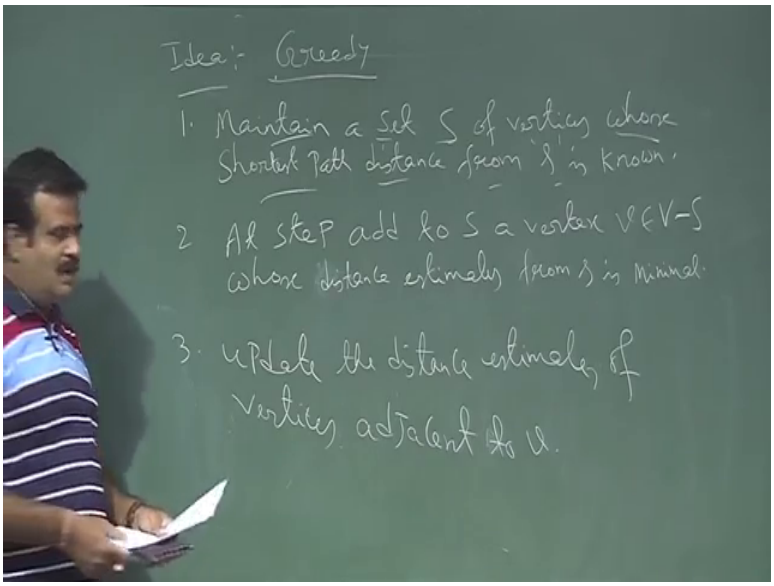
this V in S and then we have all the vertices which are connected to v. So, this is our say new u. So, we have all the vertices which are connected to this v.

So, this was having a distance estimate d of V 1, say now we have new this is capture in S. So, once it is capture in S so that means, for that vertex delta of S comma V d u basically this. So, once we capture a vertex in S for that vertex the degree will be the delta S comma u. Now these vertex which is directly connected with this vertex which we have just added in a S, now these vertices we was having one distance estimate. Now we have a distance estimate we have a path from this to this with this d u, and then we have this direct h. So, this is basically the what is the weight of this new path? 2 S to V 1. So, S to v. So, S to V 1 is basically d u plus w of u comma V 1, I mean u is this V. Now if this is greater than the path we have, if this is less than the distal estimate of V 1 V have. Then you have to update this then we got a better path.

So, this means we are having a path from S to that vertex V 1 with the cost d of V 1. Now we have a new path once we capture this in S capital S then we have a new path we can go to S to V then V to this. And that then path this d u or d V plus this and if that is that is having less weight than the earlier degree or earlier path we have, then we must have update that we must take the new path. So, that is called relaxation, we are relaxing this edge. So, relaxation means if you are updating the updating the degree of this by a new path, if it is better than the what we have early. So, that is called relaxation. So, you may have to relax this.

So, at each step will do this. So, let us write this. So, this is basically every time we are capturing a vertex in S.

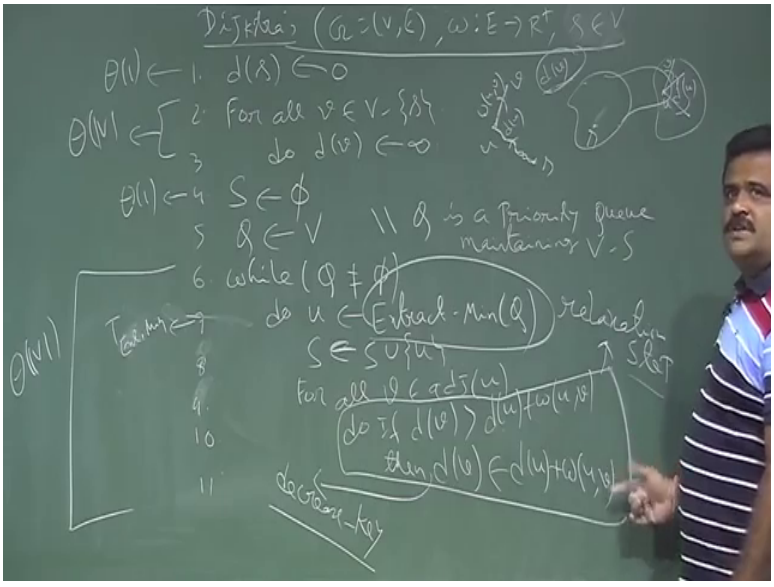
(Refer Slide Time: 12:35)



And then we update the update the distance estimate, distance estimate of the vertices which are adjacent to V distance estimates of the vertices adjacent to adjacent to v. So, this is the this is the idea of behind the distance algorithm. So now, will in a moment we write the pseudo code for distances algorithm and then we will go for a example ok.

So, let us just write the pseudo code for dusters. So, the idea is greedy. So, every time we capture a vertex in S in a greedy choice in a greedy way, because we are taking the minimum whose the we are taking the vertex whose distance is minimum.

(Refer Slide Time: 13:46)



So, that is the greedy choice. So, this is the algorithm. So, dusters algorithm. So, what are the input is a directed graph and we have a weight which is non negative. And then we have a source vertex. So, this is a single source shortest path. So, give a vertex says give a vertex says we need to find this is any vertex could be the input, but this is the vertex this is part of the input ok.

So, we are defining degree of S to V 0 this is basically giving this degrees we have maintain as a distance estimate form smallest to any other vertices. So, for all other vertex V which is this is the initialization phase do d V to infinity. This is say same as the previous algorithm this is initialization say because initially nothing has export. So, we are putting the d V to be infinity for all other vertices because we have an explore it. So that means so, that is why we are putting infinity for all other vertices. So, this is 3 4. And now it if S is initialized by infinity and then we 5 take a Q all other vertices. So, this is the this is the priority Q. Q is basically is a like we did in creeps algorithm, priority Q maintaining V minus S basically. And every time and this priority Q is this key does the degree of that vertex. And every time we extract a minimum from this Q until the Q is empty. So, this is the operation while Q is not null we do extract minimum from this Q.

So, extracting means we are deleting from the Q, and we are adding in S. This u is added with S. So, we just capture. So, we just capture a vertex in so, initially S is in this set. So, then we capture a vertex which are having the V which is a so, the this is a u which is having the minimum degree. So, we capture this in S. Then what I have to do? We have to update the distance estimate in all above the vertices which are adjacent to v. So, that is basically we need to do now. So, this is 7 this is 8. So, 9 is. So now, we need to this is 8. So, for all vertex. So, for so, after extracting for all vertex V which are in a adjacency vertex of u sorry. So, what we do? We have suppose this is this is our this is our u now this is our V 1 of the v.

So, it was having a distance estimate d V. Now this is our a somewhere here. So, we have so, this is the now if do if the degree of v. So, distance estimate of V which is having is greater than d u plus w of u v. So, what we do? We go for S to u with this path and that is the shortest path, because once we capture a vertex in S capital S that that degree will become the shortest path of S to that zone. And then it take the direct this w of u comma v. So, this is a new path from S to V and the weight of this path is basically noise d u plus this. And if this weight is less than weight of the path what we have. Then we must

update this weight we must take this new path. So, that is then d V must be updated by this new path ok.

So, this is 10 and this is 11. So, that is it. So, this is the pseudo code for detains algorithms. So, this type is called this step what we are doing is called the relaxation step, relaxation step. So, basically if the already we have some estimate distance estimate from a vertex V, v is the vertex which is adjacent to the vertex u which is just recently captured in capital. So, why did V is having some distance estimate So that means, already there is a path from S to V with the weight d V. And initialized by infinity because initially nothing has explore. Now we have now we have a path, what is that path? We go from S to u with the cost d u then plus we take this direct edge w u v. And the weight of that path is d u plus w u v. Now if the weight of this path is better than the weight you already have. Then we must relax this then you must check we must check take the new path. So, that is called relaxation. We are relaxing edge, because we got a new path for S to S to this vertex. So, that is the idea. And these we will do for all the adjacent vertices of u ok.

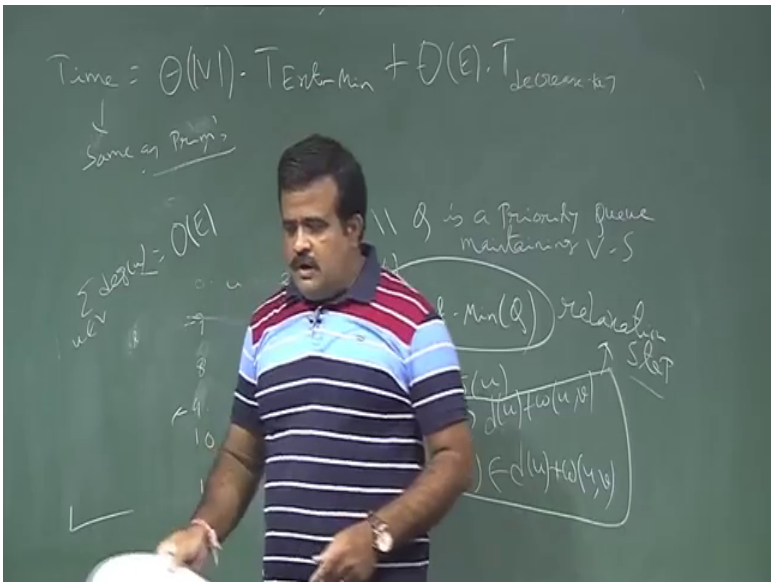
So now and this is the relaxation and this type this is basically same as decrease Q operation which we did same thing and in the preens algorithm decrease Q operation. And then the this step is called this is the extract min. So, depending on how we implement this in a data structure it will take the time. So, before going to the example what is the time complexity of this algorithm? So, time complexity is basically this is this steep is basically theta 1 and this step we are doing basically theta of V times. And this is basically these of 1 and here this Q this is the while loop and this loop is running up to V times if these it will keep on running until we deserts the vertex ok.

So, this step will be running order of V times. And inside that what we are doing? This we are doing the extract minimum. So, again depending on the priority Q how we are implementing that will if we are using a array jut we are using a vertex we are putting into a array. So, the extract minimum means we have to choose the minimum. So, minimum will take the order of n time. So, depending on this, but if we use the min. E as we discussed preens algorithm analysis if we use the min if it will take the logarithm time.

So, this is the time for this. So, time to extract minimum time to extract minimum and we have this is what we are doing time to extract time to decrease the key. So, if we up to relax this vertex then you have to go to that particular fill and have to change the degree. So, for that we need to if it is arrow we are going there we are changing. So, that is constant time, but if it is a heal if we change he did may pallet the hip of in it hip property again he need pi that. So, this is basically time to decrease key how many times this is the of this times. So, summation of time for decrease the key. So, if we just write the this time complexity of this it is same as we will see it is same as the preens algorithms times complexity.

So, basically it is just write the time complexity of this. So, so this part is time for dusters algorithm is basically order of V times extract minimum last.

(Refer Slide Time: 23:23)

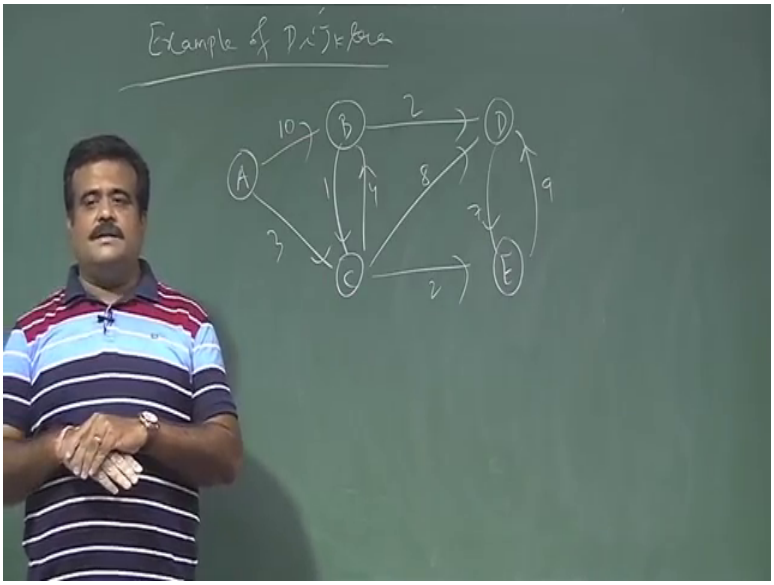


So, this is outer loop is order of V times now this is basically, summation of degree times we are doing. Now summation of summation of degrees basically this we have seen this is the anlagen order of e. So, this is basically order of E times t of decrees key operation. So, this is time complexity is same as the preens algorithm and depending on the huge data structure we are using. So, if we using if then this will be like this if we are using a array then this will be order of V and this will be constant. So, whole will be order of V square. And if we are using a if then this will be order of log v. So, this will be again

order of log v. So, depending on that. So, this will be E log v. So, any way this is the same as preens algorithm ok.

So now will talk about we will take an example for this dusters algorithm how it is working. So, let us take an example of a graph and on which we can have the dusters algorithm ok.

(Refer Slide Time: 25:25)



So, example of dusters. So, we take a graph. So, let us take this graph. So, this are the weight 3 2 1 4 8 2 7 9. So, this is given graph. And this is a directed graph. So, this is a directed graph and we are having the edge weight are non negative. So, there is no negative edge So that means, if there is a shortest path from if there is a path from u S to S is also has to be another input if there is a shortest path from S to a vertex. Then there will be a if there will be a path from S to vertex then there will be shortest path. Otherwise that S will be infinity if there is no physical path from S to that vertices. So, will continue this in the next class. We have short of time now.

So, from next class we will we will complete this example. Will work will work out dusters algorithm on this example.

Thank you.