NPTEL
NPTEL ONLINE CERTIFICATION COURSE

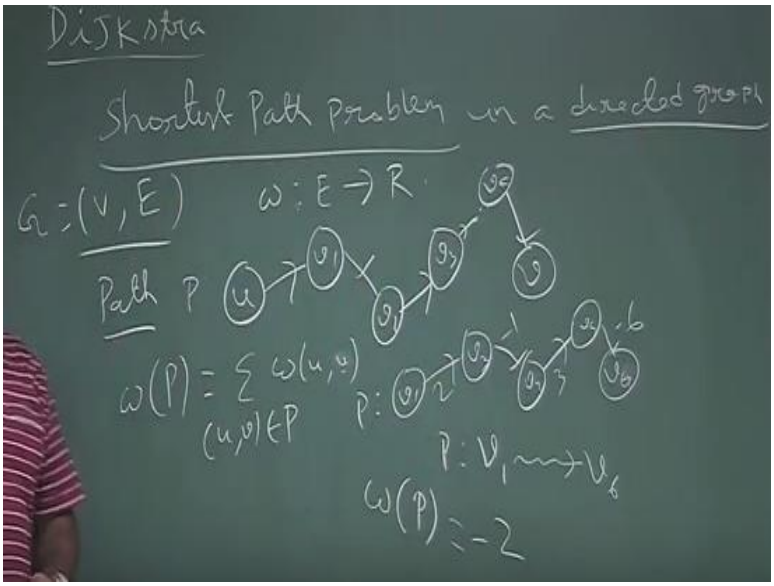
Course Name
Fundamental Algorithms:
Design and Analysis

by
Prof. Sourav Mukhpadhyay
Department of Mathematics
IIT Kharagpur

Lecture 18: Dijkstra

Okay, so we will talk about shortest path problem.

(Refer Slide Time: 00:24)



In a graph, in a directed graph, suppose we have a directed graph G(V, E) where E is the set of E is the set of vertices, E is the set of edges. And we have a H of it, so this is basically the function from A is to the real number. So these are, this is the weight function, okay.

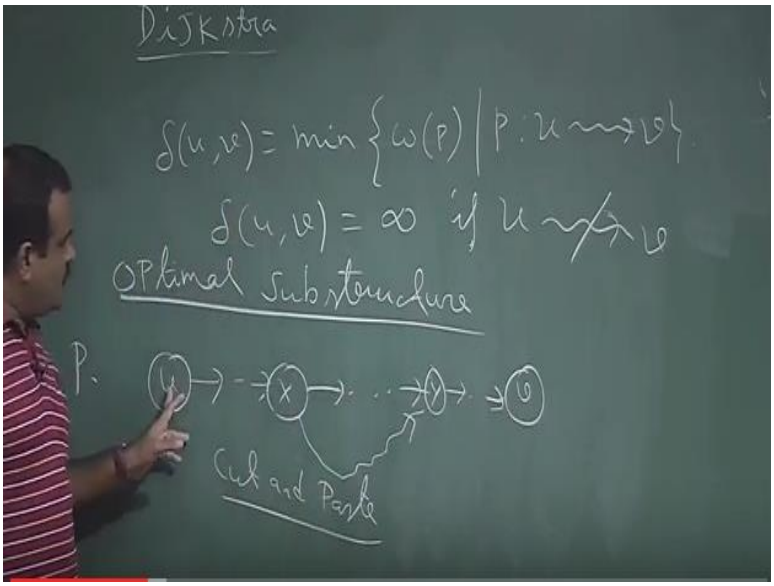
Now what do you mean by path of a graph, suppose we have, how to define path, suppose we have two nodes (u, v), now suppose we have some intermediate nodes of v_1 is the directed graph,

v_2, v_3 …. V_k and then this is, so path means a physical path, we should able to go from by follow the direction we should able to go from u to p, this is the path.

Now what is this weight of this path, weight of, this is a path p. Now weight of this path is basically this is the sum of the weight (u, v) such that this (u, v) is belongs to this path. So we just count if suppose if this is a example, suppose if u then we have v_4 and we have some intermediate vertex v_3 so this is v_3, v_4, v_5 and say this is v_6, this is v_6 and weight are said 2-1, 3-6 suppose these are the weight.

Now this is a path, this is a path from v_1 to v_6 this is v_6. Now what is the weight of this path, weight of this path is basically sum of the age weight. So this is 2-1, 3 so this is basically 5-1 and -1 and this is -2, this gives sum of the weights of this path, okay. So now we want to find out the shortest path between two nodes (u, v).

(Refer Slide Time: 03:09)



So if you denote that weight of the shortest path by δ(u, v) this is basically minimum amount all the path weight, P is the minimum amount among all the path weight, and P is the path from u to

v. So we consider all the path from into v, now whichever is the minimum so that path we take and the weight of that path is denoted by this δ(u, v).

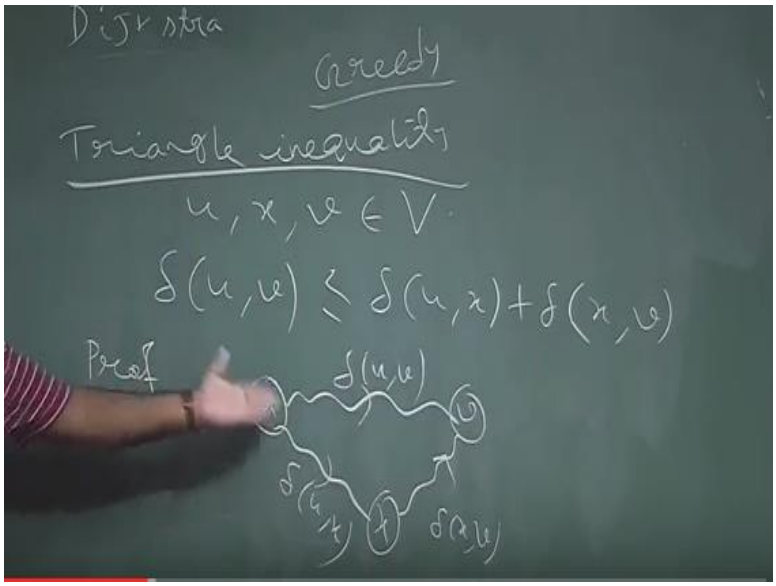
Now when, so if there is no physical path that exist from u to v then we say (u, v) is infinity. If there is no path from u to v then we say no shortest path exist, so we want to see whether optimal sub structure exist in this case or not. So that we can think for dynamic programming approach to solve this problem finding the shortest path.

So optimal sub structure means suppose we have a shortest path from u to v and in the middle we have some notes X, Y, suppose this is the shortest path from u to v. Now if you take this path from this path, from X to Y so then this will give us, this part will give us the shortest path between X to Y, if this is true for all nodes then this is called optimal.

So if not, so we can prove this, if suppose there is a path from X to Y which is shorter than this path. Then what we do this is called cut and paste technique, so what we do, we take this path a new path from u to v like this we go from u to x then we take this path then from x to this, so this is a new path from u to v and this is lesser than the path which we have.

So it contradict that, that our original path was the shortest path, because if this is a shorter path from x to y. So this is the optimal sub structure property is happening here. So we could think for a finding the, applying the dynamic programming approach. But we want to look at whether we can apply greedy approach or not.

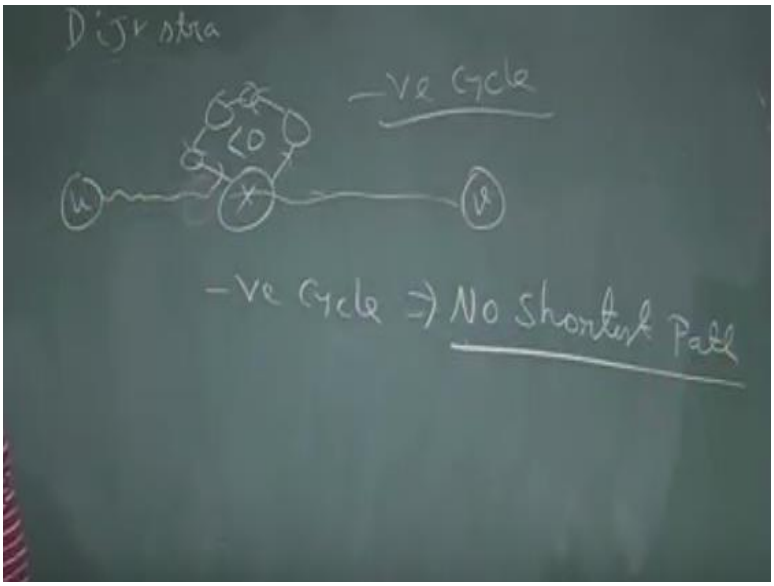
(Refer Slide Time: 06:05)



So yes we can apply the greedy one, so that is the Dijkstra algorithm we will talk about. Now that is based on inequality which is called triangular inequality. So what is a triangle inequality suppose we take 3 vertices u, x, v belongs to V now it is telling $\delta(u, v) \leq \delta(u, x) + \delta(x, v)$ so how to prove this is for any three vertex x, u, v and x so how to prove this we can prove it graphical so suppose we have this vertex u, v and x okay so this is shortest path from u to v $\delta(u, v)$ and this is the shortest path from u to x and this is the shortest path from x to v.

Okay now why this is true because see if we take this path u to x then x to v what is the weight of this path this + this so this is a path so this is this path must be weight of this path must be lesser than that so weight of the shortest path this is the weight of the shortest path so hence this is less than the another path we got a path and weight of that path $\delta(u, x) + \delta(x, v)$ so this is the proof, proof is straight forward from this picture okay so this called triangular inequality. Now we talk about when the shortest path may not excites even though there is physical path from u to v.

(Refer Slide Time: 07:59)

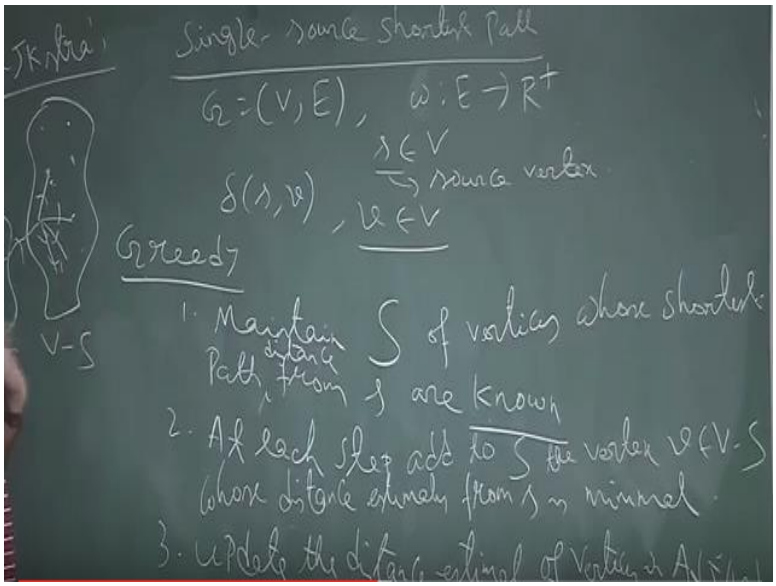


Suppose we have node u and we have a node v suppose there is a physical path from u to v okay even though there may not excites the shortest path when if suppose the in this path suppose there is two there is a negative cycle suppose this is a x intermediate node and we have a we have a negative cycle so we have cycle whose weight negative.

Negative cycle in the path then the shortest path will not excites because why now if we give me if you give me this is the shortest path I will make another loop to reduce that so we cannot find out a shortest path if the shortest path okay so negative cycle will give us will not give us the shortest path now we will talk about a algorithm which is Dijkstra's algorithm where we assume there is no negative cycle so that means to guarantee that we assume all the edge weight positive.

So no edge weight are negative so that if all the edge weight are positive there is no question of negative cycle

(Refer Slide Time: 09:32)



So this is the problem of single source shortest path problem and here so here we have given a graph G,E and we have this is Dijkstra's algorithm, we will take about Dijkstra's algorithm so for Dijkstra's algorithm we have assumption on this weight function W which is basically R+ so we are not allowing any negative weight H to effort the negative cycle Dijkstra's cannot handle the negative cycle and we have source node this is called vertex from which so this is single source from which we should able to find the shortest path from any other vertices.

So basically you are trying to find u, v where v is any vertex so this is the so these approach greedy approach so the idea is we maintain a maintain a set S is S of vertices whose shortest path from this shortest path distance δ from small is are known okay so that means this is our set S and this is the vertices V-S is basically so now if we are not allowing any negative H then the smallest must be in case because if are in s.

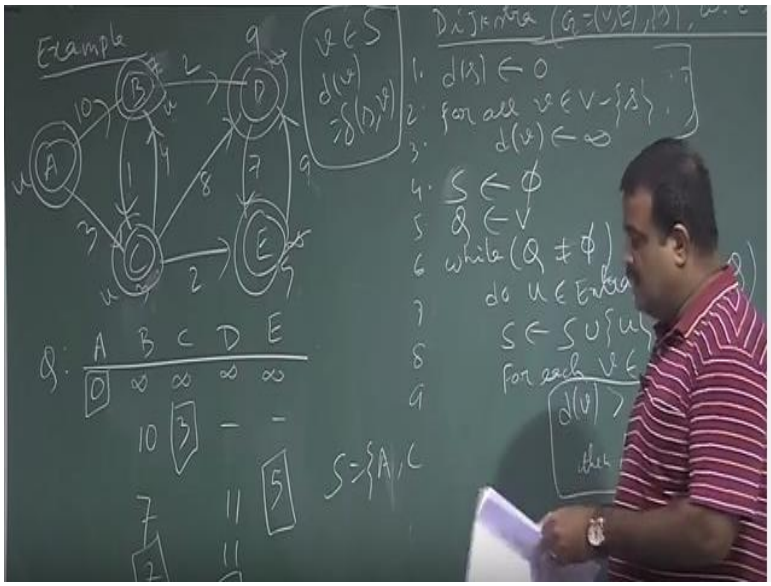
So based to it to go to the smallest is we remain in this s because we are not having negative cycle or any negative way so s that is base to S so we start S with a smallest and then slowly we capture the nodes with this so that we will write so at each state we add a node at to S the part x

formed V-X whose distinct estimate is minimum whose distant estimate from S is minimum, so that means so we have some partition over here, okay. Now this is having some distant estimate.

That means, the path from h to that node what is the cost? Now we choose the vertex V which is having distant estimate minimum among this so we capture that S V x S and then we, so V I having some adjacent vertex so now V in this now in change the addition estimate of this adjacent still stop V so that is the idea, So then in the third step we update, update the distance estimates of vertex, vertices in adjacent still are B.

We continue this until this we can empty I mean everything will be in S so that is the idea so let us write the code, so this is the greedy choice which is the minimum we are taking that, so that is the greedy approach so let us write the pseudo code for this then we will take an example to execute that port on that.

(Refer Slide Time: 14:36)



So this is the pseudo code for Dijkstra's algorithm so input is a graph and a source packets and a H vector which is, okay. So what we do? We just put the degree of this source vertices as 0 and the remaining degree is infinity so that means the distant estimate is that goes vertex and not yet

explore, so for all V belongs to V-A we put the distant estimate of third vertex as infinity, okay. Now we initialize this by empty and now we take a priority queue, ESP which is a priority queue which contain all the elements of V-A and then slowly we will extract the, this minimum distant estimate.

So we extract the node which is having the minimum degree so and this will continue until while Q is not empty, so this Q is a priority Q, okay. So while Q is not empty, what we do? We extract the minimum element from this Q this we can implement either in array or by F we know the F priority Q implementation, so now this we are going to add in S, okay. So now we will change all the adjacent vertex of S for each all the adjacent vertex of U.

So this U is now added in now we check all the adjacent vertex of UY suppose V is the adjacent vertex of U, now it was having a distant estimate, now it we check that dv for each adjacent vertex of U, you check the dv the distant estimate which it was having so that means there is a path from S to this node, now we got a new path you can conform S to this node by du then we can take the direct path.

If that is better than we replace that by this, so if dv is greater than du+ w uv this is the new path so that means we can conform S to this is S source node, S to U then we can take this direct path, if that gives us better distance to be that earlier then we should replace that, then dv will be replaced by du + w uv, so this is what is called this step this is called the relaxation step we are relaxing the relaxation state, okay.

And this is also called the decrease key this we need when you talk about the time complexity how we can go there and change it. So this is the code, this is the pseudo code for the extras algorithm and this type is the relaxation step, we will try to relax each of the verities which is adjacent to u. okay, so let us take an example, let us take a graph so suppose A,B,C,D,E so this is the connection so this is the directed graph okay, so this is direct graph and we have to give the weight and we are not allowing any negative weight to avoid the negative cycle 2,8,7,9,2 suppose this is the graph and we have to execute this discrete algorithm on this, okay.

Now suppose we choose this as a source vertex, so this is the priority Q, so A, B, C, D, E, so we are keeping the distant degree this is the source vertex 0 other side infinity that is what we are doing in the initialization step, this is the initialization step we are keeping degree of this source vertex is 0 and remaining are infinity. Okay, so now S is initialized by empty and this everything we put into the priority Q. Now we extracted the minimum, so minimum is basically this one A, so we put A in S, so this is out u, so this is our u now, now we execute this so we check all the adjacent least of Q so this is in adjacent at least, so this v adjacent at least of u, so this was having the degree, this was having the degree infinity.

So now this will be replaced by, so this 10 and this will be replace by 3, okay so this B is now, B is now 10 and E is now 3 and this will be remain unchanged, okay. So now this will be remain unchanged means these are infinity, now we take the extract the next minimum so that is 3, so that is basically C, okay C now. Now this is our u now, now this is our u now who are the v, v is the adjacent so this is D now we check the degree of this it is infinity it was infinity now we have a path from A is 2 this node by A is 2 this node by 3, and the we have this 2 this will be now 5, okay.

So that means this will be E is now 5 and this is also now this will be 11, this is D E is now 11 and now we have this path so it was having a discrete estimate thing that means there is also path from A is 2 this node with a costing. Now we have a new path what is that, we can go from A is 2 this node by cost 3 and then we can take this direct path 4 so with the cost 7, so 7 is greater than 10, so we can replace that 7, okay.

So now this v is now 7, and these two are already in s so this is C, now we take the next minimum so this is the 5 that is basically E, so now we check the adjacent list of this so this is now v is this so now this was having a distance estimate 11, so that means there is a path from A is 2 this node with the cost 11. Now we have another path we can go to A is 2 this node with the cost 5 and then we can the direct path 9, so 5+9 14 which is not good as 11, so we keep the earlier path which is better.

So we are not relaxing this node H, we are not relaxing this node, okay. So now which is the minimum, so this will be remain and so 7 and 11 will be remain and same, so now 7 is the minimum, so 7 is the minimum, now we take this is now u, now we take all the adjacency list of this, so this one adjacency vertex D, so now it was having distance estimate D(n) now we have a new path, so we can go from A is to this node we take our 7 and then from here we can take this direct path 2 so 9.

So 9 is better so this will be replace by 9, so D is now 9, okay and the remaining one, so we have this one we can just have a cross reply so this is having distance estimate 3 which is already in here so we do not need to check that but to be cost by 5, so now we have a new path, we can go from A is to this node with a cost 7 then we can take the direct path 1, so 7+1 8 which is not good at C.

So whoever is the node in A then the distance estimate is done, so if, if a node, if a node v is in S that means their degree is basically ḍ s, v this is the correct place, so once we capture these nodes in a then that degree is becoming the minimum these to estimate from s, s2 that note. So then we take 9, so 9 we can just have a verify whether this is better be yeah so then all done, and these are basically Δ's, this boxes are basically ḍ S, that notes so here we are not allowing any negative cycle.

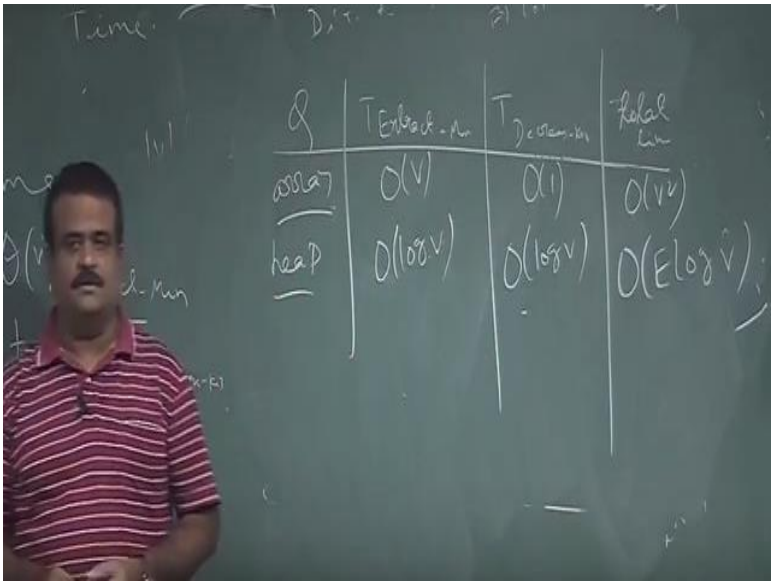
To avoid, yeah to avoid the negative cycle we are not alloying any negative weight h, so now what is the time complexity for this, what is the runtime for this, so run to calculate the runtime we have to check, to calculate the runtime okay, so how to calculate the runtime for this, so time complexity for this, so time will depend how we are use implementing this pyritic Q, whether in array or the u so basically this is, this is of order of v times.

This loop because this is about our v and then this is also about our v, and the inner loop, so this loop the for each of this adjacent series this loop is basically, this is the degree of u degree of u basically so if we take the sum of this degree, degree of u this is basically give us order of E by hand setting Lamar, this can be prove so some of the degree is order of v so basically what we are doing, here we are extracting the minimum that cost we do not know.

It depends on which data structure using we are using, we using a array or if and then we are decrease the p so if the relaxation is done then we are going to that particular note and we are decreasing the, we are changing the degree so that is sort of decreasing the key. So the time is basically same as time per primes algorithm so time is basically order of v, I mean size of v into the time required for extract the minimum last order of E and the time require for decrease the key.

Okay, so now if we depending on which data structure will use for our pyritic q the time will be depends on that, so let us just slide that.

(Refer Slide Time: 27:23)



So this is the runtime so now if we implement the pyritic Q using say array so this is the time for extract the mean and this is the time for decrease the key, and this is the total time for this time, total time for extra, so if we just use a array for array to have this pyritic Q we are putting everything into the array.

Then how much time we will take for extract the mean, so minimum we will take we need to read the all the element so this is basically order of v and decrease the key is just we go to that particular position and we change the value, this is basically constant time and send then this will be just basically, okay. Now if we use the binary heap mean then the decrease the key we will take extract the mean will take because minimum is the in the root so if we are using meaning so we will just take it that and we the heap the heap property is highlighting, we need property.

So we need to have the mean heapify, so that will take basically order of log v and this is also order of log v, so this is basically order of v log v, okay, so this is the time complexity for Dijkstra algorithm, thank you.