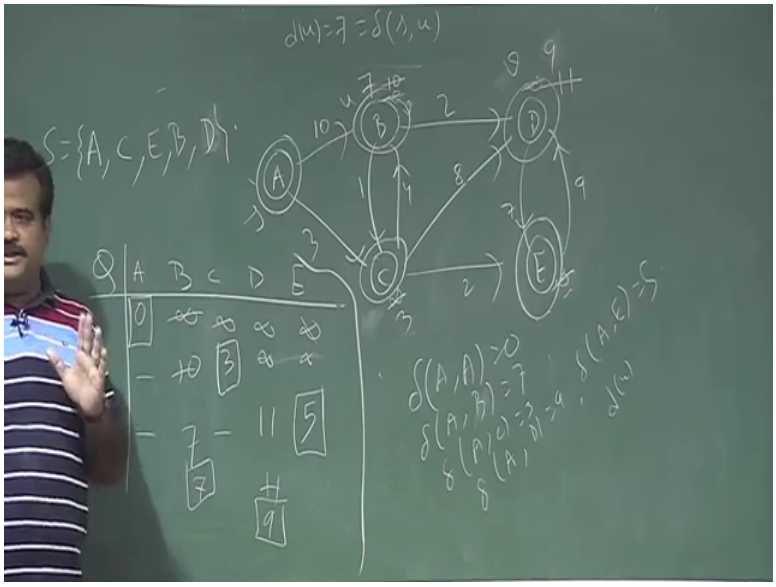
**An Introduction to Algorithms**
**Prof. Sourav Mukhopadhyay**
**Department of Mathematics**
**Indian Institute of Technology, Kharagpur**

**Lecture - 44**
**Example Of Dijktra**

We are talking about Dijktra's algorithm. So, we want to work out the Dijktra's algorithm on this given input.

(Refer Slide Time: 00:17)



So, just to recap in the Dijktra's algorithm what we are doing we are starting with the vertex s that is also a part of the input suppose this is the graph and suppose A is our s ok, so this is our s say starting vertex this is also part of the input. So, the input will be a vertex which we have to go; this is single source shortest path problem. So, given a vertex s we have to find out the shortest path from all the vertices, ok.

So, basically what we are doing we are just maintaining this thing, we are putting everything in the Q. So, our Q is basically A B C D E. So, we are putting everything into the Q. And we are putting the degree of this as this is the initialization step of this Dijktra's algorithm if you recall. So, you we are just putting the degree of s as 0 and for all other vertices v minus s we are putting degree to a infinity, because if this is a initialization step nothing as explore. And this Q this is the basically all the vertices. So,

we are putting all the vertices s is empty capital S this m p initially and we are putting all other vertices in Q which is the priority Q, ok.

So, basically the idea is we just capture one; each time we just capture one vertex in s as we update the distain estimate of that vertex which are adjacent to that vertex which have recently captured. Now, what we do we do the extract minimum. So, we do the extract minimum u use the vertex. So, extract minimum from the Q. Now, everything is infinity accept this. So, this will be our u and this once we extract minimum will put it in S. So, S will be S union. So, S will be A because it was empty. So, this is our u. Now once we have u then. So, we look at all the vertices which are adjacent to u. So, this is one vertex which is adjacent to u. So, this is v vertex, so all the degree was infinity. So, we look at the vertex which are adjacent to u.

Now, its degree was infinity, now we have d u d is basically 0; so 0 plus this w u comma v which is 10. So, that is basically 0 plus 10 is 10. So, 10 is greater than infinity. So, we have to decrease the key we have to change the degree, so these become 10. Now this become 10 now. Similarly now this is v which is another adjacency vertex of u. Now it was having a degree infinity, now we have a path we can go from s to s with the cost 0 then we can take 3. So, this degree will be now. So, this is the relaxation we are doing; we are relaxing the vertex. So, if you do this relaxation those c became now 3.

And then they are all the two adjacency vertex of this and these two will be copied no change over here. So, you can just this one copy this one ok. So, now this is the first round then again we have to check whether Q is empty Q is not empty still they are they are in the Q. So, Q means d minus a S capital S. So, they are in the q, so this two are infinity and this is 3. Now you have to extract minimum again. So, now, if we extract minimum, so then this 3 is the minimum because now all are 10 3 infinity; infinity 3 is the minimum. So, this is our u now. So, u is basically extract minimum from the Q.

Now this 3 is the minimum. So, now, this u is now added so u is basically c so c is added in s. Now, this is our u, now we check all the adjacency vertex of this. So, this is one of the adjacency vertex this is v now this is this is having a degree infinity now we have a path from s to this node how. So, this d u is basically 3. So, that is same as delta of s comma v. That means, once a vertex is in s we you are capturing is vertex. So, once u in

s that means, d v is basically delta of s comma u. So, this is the correctness of this theorem Dijktra's theorem

So, once we capture a vertex in s then and this correctness is coming from triangular inequality. So, once we capture a vertex in capital S for that vertex the delta of the degree is basically becoming the shortest path from s to that vertex. So, now, this is 3. So, now, from s to v what we can do; we can go from s to u with a cost 3 then we can take this direct path, so 3 plus 2 5 which is greater than infinity. So, we change this to 5. So, e is now became 5. Now it is another this is also v, so it was infinity now we have 3 plus 8 11; so is 11. So, d is 11. And now this is another vertex which is adjacent to this now this is our v now it was having a degree 10. That means there was a path from s to this vertex with a cost 10.

Now, we have a new path from s to this vertex how; we can go from s to u with a cost 3 then we can take this direct path 4; so 3 plus 4 7. Now, 7 is greater than 3. So, this path is better, so you must relax this h. So, you must relax this h. So, this will be the now 7. So, this 10 now became 7. So, this there is no other adjacency vertex of u. So, what now we do what extract the next minimum, next minimum is 3 a 5. So, 5 is basically this one, so this is our u now, we capture this in s. Now, this is our u, now what is the v? V is this one there is only one adjacency vertex of this, this is v. So, what was the degree of v? Degree of v was 11.

That means there was a path there is a path from s to v with a cost 11. Now we have a new path what is that path. So, we can go from s to u and this cost is basically d u which is basically 5, because this has been added in s recently. That means, the degree is becoming the delta for this vertex. So, this is d u and then we can take a direct path that is basically 9. So, this cost is 14. So, we come from s to this with this shortest path, then we take this direct h which is 9 so 5 plus 9 14 which is not good as 11. So, we will not relax this vertex, because the earlier path is better than the path currently now we are having. So, we are not going to relax that vertex. So, this we remain 11.

So, this is the situation. So, this will be the infinity infinity and this will remain 11 and this will remain 7. Now we extract the next minimum, next minimum is 7. So, this is our u. So, this is 7 and this v we will be captured in s. Now, this is our u which is 7. Now, delta of u is basically 7 which is basically d of v is 7 delta of s comma u. That means,

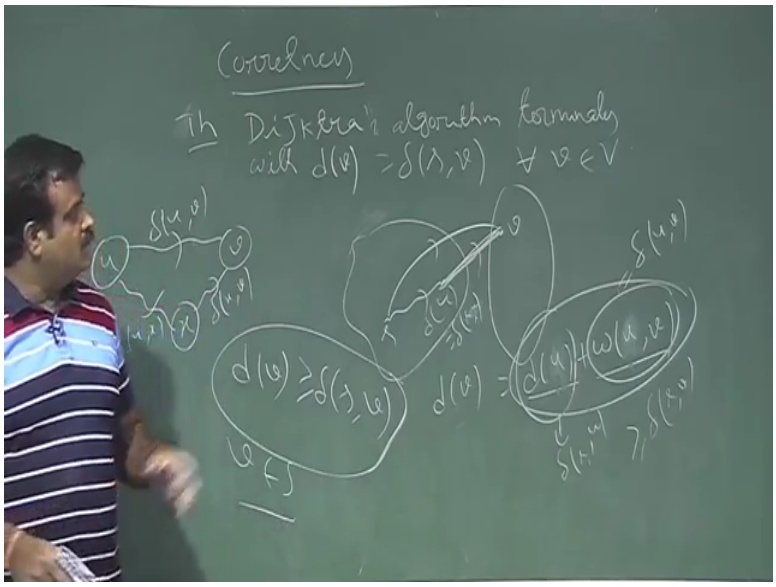
from s we have a path and that is a shortest path we have a path from s to u with a cost 7. So, that path is basically shortest path because this u is recently captured in s. So, whatever the vertex we are taking this is for that degree will become the shortest path weight from s to that vertex, ok.

So, now let us consider all the adjacency vertices of this node. So, this is one adjacency vertex. So, this is now v, so if this is v then the degree of v was 11 now we have a new path we can go from s to u somewhere and that path is basically 7 cost. Then we can take this direct path 9 and that is better than 11. So, new path is better than 11. So, you have to relax this h. So, we have to change this to 7; sorry 9 7 plus 2 9. So, this 11 now became 9.

And is there any other adjacency vertex of this? Here this is one adjacency vertex of this, but this is already captured in s, but again we can cross verify because we are getting a another path from this is also a v which is adjacency vertex of this, now we are getting another path from s to v like we go to s to u some way that will cost of 7 plus this one; so 8. But we already have a path we cost 3 which is 8 is not better than 3. So, that is a cross verify that is it working fine. So, that is it.

So, this is the two vertices which will adjacent to this then we again call this extract min and this is the vertex which are going to be add in D s. So, that is it. And this Q is empty now because all the vertices are now deleted from this Q priority Q and has been captured in s. So, now, these are basically deltas. So, deltas of this box are basically giving us the delta. So, delta of s is basically a delta of a comma a 0 delta of a comma b- b is basically 7, delta of a comma c is basically 3, delta of a comma d is basically 9, and delta of a comma e is basically 5. So, these are basically degree of those vertices. So, this is the Dijktra's algorithm execution.

So, here we are assuming no negative cycle we cannot Dijktra's cannot handle the negative cycle. So, that is why you strictly follow there will be no negative weight h in order to avoid the negative cycle. So, just let us go for a quick correctness of this theorem. So, the idea is we just grow this capital S, we capture each step, we capture one vertex in capital S which distain estimate is minimum. So, distain estimate from small s to that vertex is minimum and that is the greedy choice. So, that is why this algorithm is a greedy algorithm ok.




So, correctness is basically telling us. So, there is a two theorem for this correctness. So, it is telling us I mean. So, we can just write in one theorem. So, Dijktra's algorithm terminates, so when you finish this execution then we d v is equal to delta of s comma v for all v. So, that is what we are expecting, because when we capture a vertex in s. So, how to prove we keep the outline of this proof? So, what basically when we execute when we finish the execution of this algorithm then the degree is becoming this. So, to prove this- we can just proof this. So, if you recall the Dijktra's algorithm what we are doing we are maintaining a set s. So, first of all we are initializing each degree by; so we are initializing degree of s is 0 and degree of all other vertex this is the initiation step- all are vertex from v comma s v comma s we put the degree to be infinity, ok

Now, in the first step the s capital S is becoming I mean we have just capturing the smallest in capital S because. Now, small s this degree is 0. Now, what is the delta of s comma s? Delta of s comma s is 0 why, because there is no negative cycle. That means, from s to s what is the best for we can go from a to s. If you remain that s because there is no other way if you start from s if you there is no other way you can come back to s with less than 0, because there is no negative cycle. If there is negative cycle then there could be a chance. There we can have this hope in this path and then we can just do that. So, since there is no negative cycle that means, there will be no question of reducing this to further 0. That means, this s will be in s and the degree of this is 0. So, for s we are we are fine. So, d of delta of s comma s is basically 0, ok.

Now what we are doing? We are doing here. So, we have s at capital S which is small s should be there, now we have a set v minus s. Now there are some vertices over here, now, among these vertices. So, these are the degree of these vertices, this is u 1, u 2 suppose u k I mean there are say. Now we are what we are doing o once we capture a vertex in u then we take a direct we take a v over here which is basically, so this is basically delta of. So, what is the degree of v? So, degree of v is basically degree of u plus delta of u comma v. So, initially this is s basically u is s,ok.

So, now if we choose the minimum among this, this will be become. So this is basically, if you take s to this vertex any other path, any other path from s to that vertex that will be the less than that. So, my claim is delta of v must be greater than equal to sorry d of v must be greater than equal to delta of s comma v. So, how to prove this? So, to prove this what we do? So, this is the shortest path from s to v, now this is one path from s to u then u to this. So, this is the weight of that path. Now so delta must be less than equal to that. So, that is the triangular inequality we are using. If you remember the triangular inequality if we have two vertex u v and if we have another vertex h then if we have a shortest path from s to, so this is basically delta of u comma v and delta of u comma x and this is basically delta of x comma v.

Now, this is basically s, u is basically s now this is basically say s u is initially this u is s because initially you are capturing s in that set. So now, this is basically the shortest path from s to s, because this is degree which is 0 and this is basically the direct h from s to that vertex or any; sorry we are not talking about s anyway let us take any other vertex. So, this suppose this is two for we can use by index; suppose this is true for up to k now in the k plus one step we are capturing one node and we have to prove that is true for k plus one step.

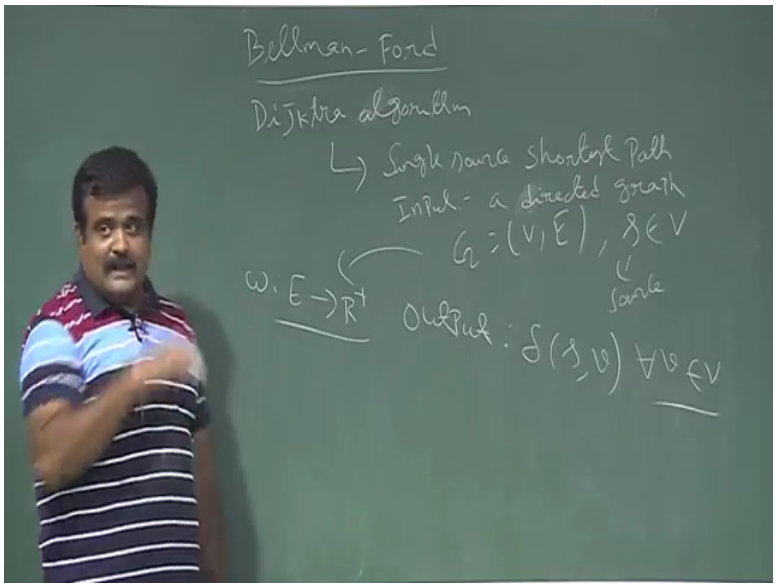
So, if we assume this is true for up to k step so that is the degree. So, this is basically delta of s comma u, because this is the assumption we are making. So, this is basically delta of sorry this is d now this is d. So, delta of s comma u by the index of this is and this is the direct path. So, this is the shortest path from u to v. So, this is basically must be this, this must be less than the shortest path the shortest path must be less than this now if this happened to be the minimum which we already have this has to be there. So, that is by the, because this is the delta of u comma v. So, this is delta of s comma s. Then

this means this must be greater than equal to delta of s comma v. So, this is coming from this triangular inequality.

So, when this is true if the s is in if the v is in capital S then the equality will occur. Any way this is the correctness of this theorem and this is coming from the triangular inequality. And we have already seen the time complexity for Dijktra's algorithm which is basically same as spins algorithm. So, the analysis part of Dijktra's is same as spins algorithm.

So, now will talk about another algorithm which is basically can handle the negative weight cycle which is called Bellman-Ford algorithm.

(Refer Slide Time: 23:18)

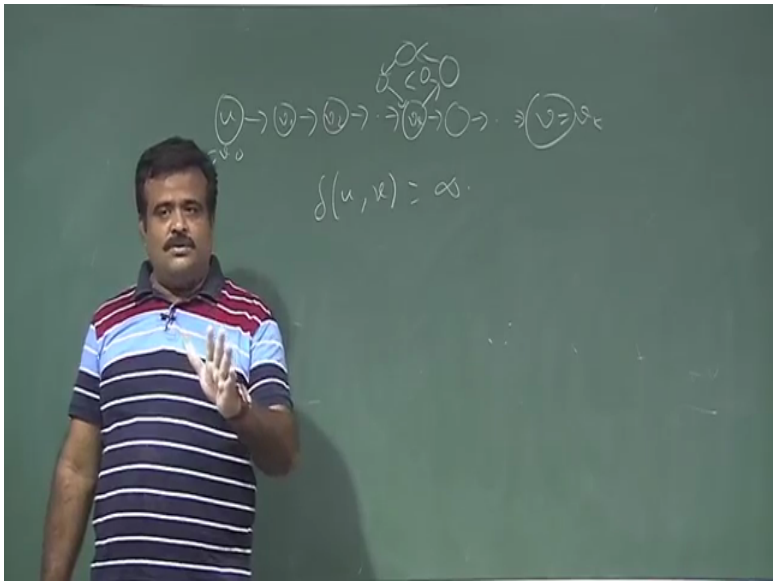



So, before going to the Bellman-Ford algorithm let us recap the problem with the negative weight cycle. So, for we have seen the Dijktra's algorithm where in Dijktra's algorithm what we have; so this single source shortest path problem single source shortest path.

That means, we have given a input is a graph, we pay vertex as a source a graph a directed graph g V comma E and a vertex which is source vertex and the output will be the delta of s comma v for all v belongs to capital V. And also we have the weight function E to R plus. So, here we are not allowing the negative weight h because Dijktra's cannot handle the negative cycle; so no negative cycles.

So, we have to guarantee that in order to run the Dijktra's algorithm we have to guarantee that there is no negative cycle to guarantee that we are taking this side we are taking no negative weight h. So, that that will ensure that there will be no negative cycle. So, what is the problem with negative cycle; if we have a negative cycle then all though if there is a path from u to v there may not have a shortest path.

(Refer Slide Time: 25:22)



If we have vertex u to v, so this is say v 0, v 1, v 2 say v l v k and if there is a negative cycle over here, ok; which will not the case in Dijktra's algorithm because there is no negative weightage, but in general we can have a negative weightage so that may create a negative cycle. So, if there is a negative cycle then there will be no shortest path.

So, in that case delta of s comma v will be infinity, because if we cannot claim that there is a shortest path because if we claim this is a shortest path then will have a analogue loop in this cycle. So, this will reduce further. So, this way we can keep on reducing further. So, this way we cannot have a shortest path from u to v if there is a negative weight cycle.

So, the algorithm which will discuss in the next class is called Bellman-Ford algorithm. And that Bellman-Ford algorithm can handle the negative weight cycle. So, for Bellman-Ford algorithm or graphics any graph with the any h weight. So, we are allowing the negative weightage also, so there is there will be chances of negative cycle. So, Bellman-Ford can detect the negative cycle. If there is a negative cycle there will be no shortest

path, but Dijktra’s cannot say that there is no shortest path- since there is negative cycle Dijktra’s cannot say that, because Dijktra’s cannot handle the negative cycle. So, that is why we take the assumption the weights are positive. So, to in order to avoid the negative cycle, but in Bellman-Ford it is a general algorithm. So, it is more general than Dijktra’s. So, there we are allowing the negative weight h, so there could be a negative cycle. So, Bellman-Ford should able to tell us that there is no shortest path because of negative cycle.

Any way we will discuss this Bellman-Ford algorithm in a next class.

Thank you.