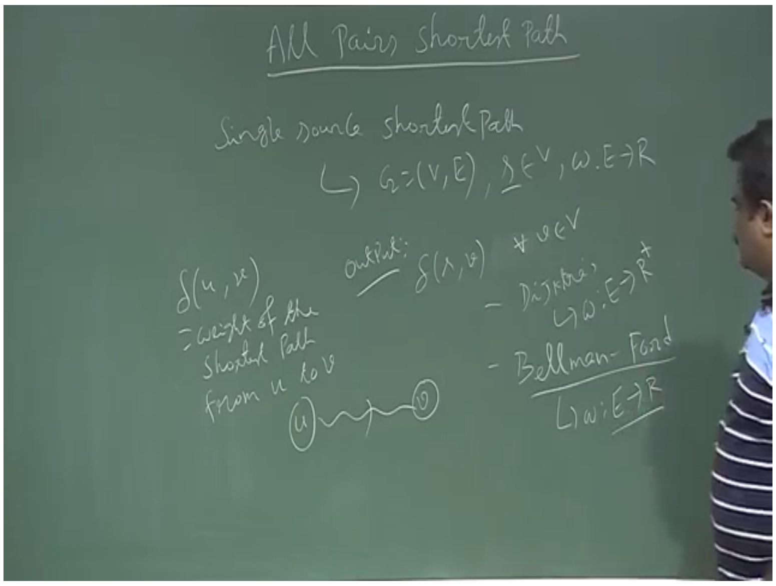
An Introduction to Algorithms
Prof. Sourav Mukhopadhyay
Department of Mathematics
Indian Institute of Technology, Kharagpur

Lecture - 48
All Pairs of Shortest Path

So we talk about. So, far we have seen the single source shortest path so; that means, you have given a input graph directed graph and so what are the single source shortest path.

(Refer Slide Time: 00:29)



All Pairs Shortest Path

Single source shortest Path

$\hookrightarrow G=(V,E),\ \underline{s}\in V,\ \omega: E\to R$

Output: $\delta(s,v)\ \forall v\in V$

– Dijkstra's
$\hookrightarrow \omega: E\to R^+$

– $\underline{\text{Bellman-Ford}}$
$\hookrightarrow \omega: \underline{E\to R}$

$\delta(u,v)$
= weight of the Shortest Path from u to v

$$\text{(u)} \rightsquigarrow \text{(v)}$$

So, single source means input is we have a directed graph G which is V comma E and we have a source vertex from where we need to get the shortest path weight of the shortest path from other vertices. So, that is also part of the input, that is given and we have the weight function E to R and. So, this is and the output is basically delta of S comma V which is basically weight of the. So, delta is basically. So, delta of u comma v is nothing, but weight of the shortest path from path from u to v.

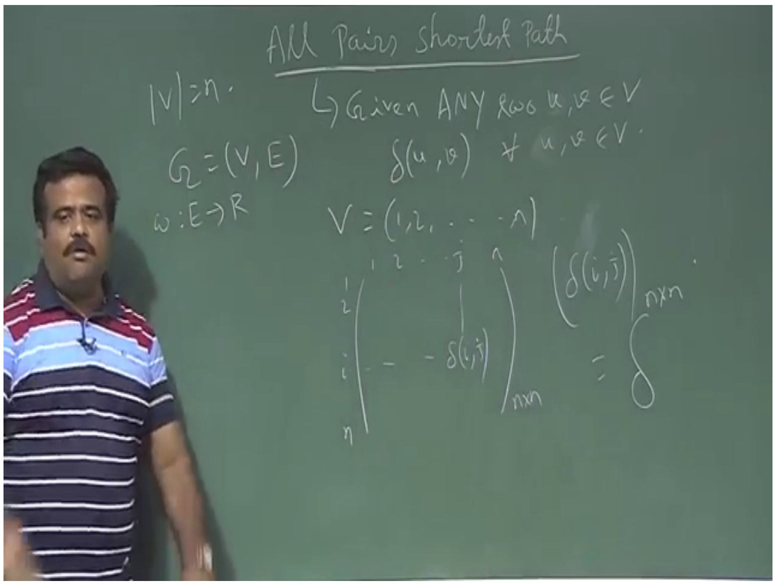
If it is exists then that that will be the weight otherwise if it is not exist then it will be infinity. So, that is basically we have a vertex v we have vertex u and v. So, you consider all the paths forms from u to v if there is a path I mean shortest path we will not exist if there is no physical path and also if there is a physical path it may not exists, that is a negative cycle of path which we have discussed earlier.

Now so shortest path means. So, we take all the paths among this all the paths whichever is the giving us the minimum weight. So, that is basically shortest path from u to v and weight of that path is denoted by delta u v and it is infinity if there is no way weight that will not exists. So now, this output of this single source shortest path algorithm is basically this is for delta is for all either infinity or it will give us some value. Now we have seen the two algorithms for this,; one is Bellman Dijkstras algorithm, but this algorithm has a limitation that it cannot handle the negative cycle. So, to ensure that we are taking that we are not allowing any negative or weights are non negative. This is to ensure that there will be no negative cycle because Dijkstras cannot handle negative cycle.

But we have generalized algorithm which is more ensure that Dijkstras which is Bellman Ford and Bellman Ford, we have no restriction on the edge weight weight because Bellman Ford can handle the negative cycle. So, Bellman Ford is a general algorithm where we have no restriction of edge weight weight. So, this is a weight is a function of E to R. So, you may have the negative weight so; that means, there is a chance of negative cycle. So, any algorithm Bellman Ford algorithm can handle that. It can report there is a negative cycle in the graph.

So, these are the single source shortest path; given a source is part of the input you have to find weight of the shortest path from that source to any other all the vertices. So, that was the single source shortest path. Now today we will discuss all pair shortest path problem. So that means, given any two vertex we should able to find deltas. So, all pair means given any two vertex u v, we should able to find out delta of u v and this is for all vertex all the vertex u comma v.

(Refer Slide Time: 04:22)

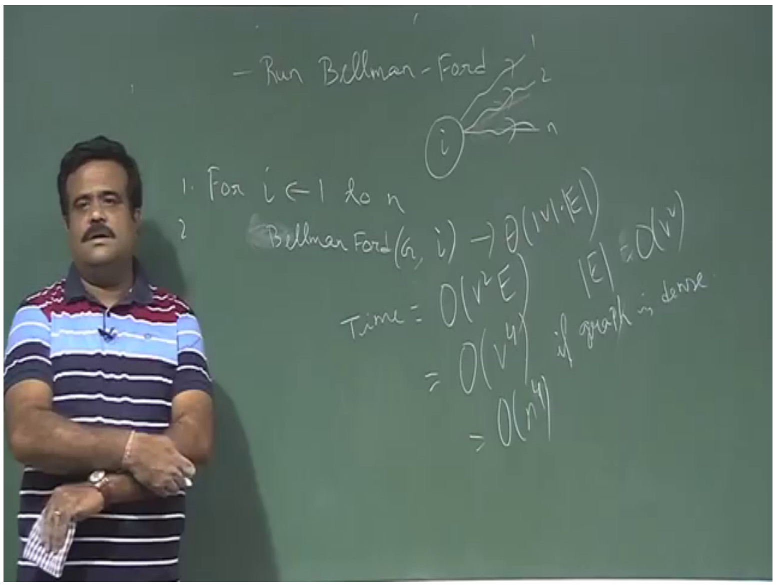


So, all pair of vertex basically. So, that is the problem today. So, this is called all pair shortest path problem. So, given any two vertex u v we should able to find this. So, basically this is if we write this in the matrix from. So, basically suppose V is say suppose there are n vertex. So, suppose there are n vertices. So, this is our graph V comma E, suppose order of V is n and so this input of this is also a graph and we have a edge weight R we are taking the general case, yeah I mean if we want to apply Dijkstras then we have to restrict our self on V to R plus. Suppose this are all vertices v1, v 2, v n. So, for simplicity you just write1 2 up to n, v 1 to v n.

So, basically what we need to find out? We need to find out this matrix. So, this is a v 1, v 2, v n this is v 1, v 2. So, say v i, v n this say j. So, what is the i j'th element of this? it is basically delta of i j. So, this is a n cross n matrix. So, basically we need to find out delta of i j, this matrix we need to find out. So, this matrix we denote by big delta. So, this is basically all pair shortest path, given any two vertex v 1, v i, v j we should able to get this deltas. Delta could be infinity if there is no shortest path from v i to v j, but this is the problem; problem is to find all pair shortest path. So, what is the input? Input is a graph with a weight edge, weight with a edge weight and there is no single source giving in because this is a all pair shortest path and the output will be the this matrix delta. So, how we can do this?

So, this we have to find out this deltas and this. So, how to do this? We can we know the Bellman Ford algorithm single source shortest path.

(Refer Slide Time: 07:20)



– Run Bellman-Ford
1 2 n
i
1. For $i \leftarrow 1$ to n
2 Bellman Ford(G, i) $\rightarrow \Theta(|V| \cdot |E|)$
Time = $O(V^2 E)$
$|E| = O(V^2)$
$= O(V^4)$ if graph is dense
$= O(n^4)$

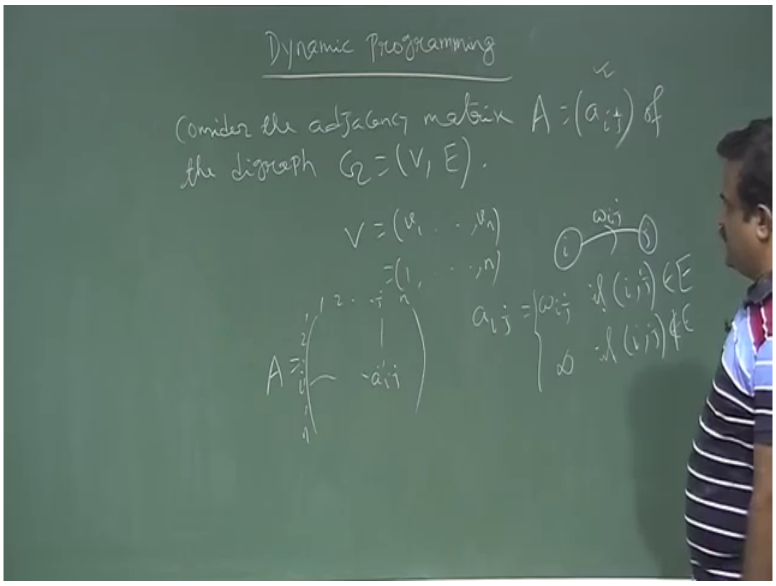
So, we can run this Bellman Ford. So, the first idea is we can make use of we run Bellman Ford for all pair of vertexes. So, basically we take any two vertexes. So, first take, so we fix the i and then from fixing. So, basically we need to find this deltas no. So, we fix i and then from this v i, we find all the delta of any other vertices. So, these are the path, shortest path. So, 1, 2, v 1, v 2, say v n, delta of i comma v 1, i comma v 2 delta of i comma v n. So, this is basically we run the Bellman Ford for all such i.

Now, what is the time complexity? So, Bellman Ford each time Bellman Ford we will take time from time order of V into E, size of V into E. Now this is we are running for. So, this is basically if you write the code for i is equal to1 to n; run. So, Bellman Ford, on this graph with this i as a source vertex that is it; so v i as a source vertex. So, what is that? So, these will take order of V into E and this is order of V times. So, the total time complexity for this all pair shortest path this order of V square into e or in size of. So, this is if this is a dense graph.

If the graph is dense graph then order of E is basically order of V square. So, this is basically order of V 4, if the graph is dense. So, if the graph is dense this is so; that means, this is a order of n 4, n to the power of 4, n is the size of V, but we want to do something better than that we want to do it in order of n cube. So, now, we want to see whether we can use the dynamic programming technique for this. So, you want to see whether dynamic programming technique can be used to solve this problem.

Programming dynamic programming technique; for finding the all pair shortest path problem.

(Refer Slide Time: 10:16)



Dynamic Programming

Consider the adjacency matrix $A = (a_{ij})$ of the digraph $G = (V, E)$.

$V = (v_1 \ldots, v_n)$
$= (1, \ldots, n)$

$$A = \begin{array}{c|ccccc} & 1 & 2 & \cdots & j & \cdots n \\ 1 & & & & & \\ 2 & & & & & \\ i & & & & a_{ij} & \\ n & & & & & \end{array}$$

$$\textcircled{i} \xrightarrow{\omega_{ij}} \textcircled{j}$$

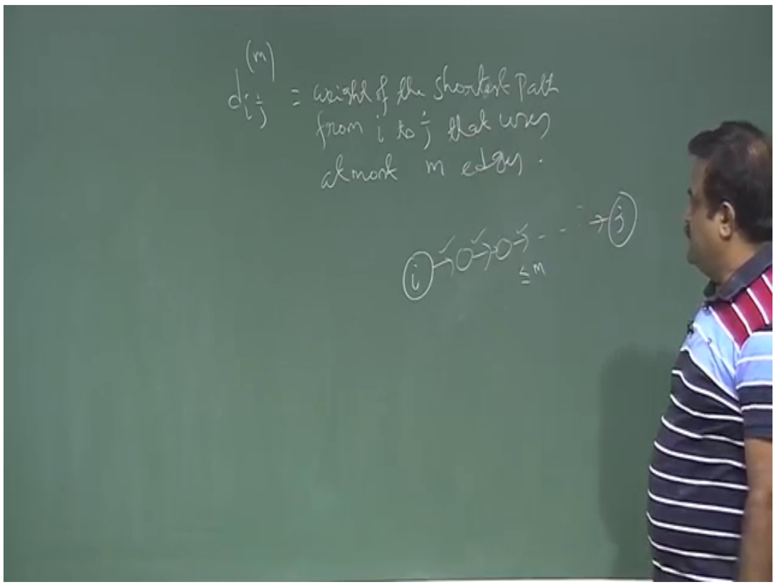
$$a_{ij} = \begin{cases} \omega_{ij} & \text{if } (i,j) \in E \\ \infty & \text{if } (i,j) \notin E \end{cases}$$

Because you have seen the shortest path is having hallmark for dynamic programming like we have seen the in the previous classes that it is having the optimal substructure problem and also the repetition of the sub problems. So, there is a indication that those are the hallmark for driving. So, there is a indication that we must go for we can think of dynamic programming problem.

So, let us just talk about the dynamic programming technique for this. So, consider the adjacency matrix A. So, A is basically a i j. So, of the diagraph, so we are given a diagraph or directed graph the diagraph G V comma E. So, basically what we have? So, A is basically what? A is basically the adjacency matrix of. So, if you take two vertex of V is basically v 1, v 2, v n. So, there are n vertices say. So, for the simplicity we are taking1 to n.

Now, v i g is basically the matrix, this is a matrix adjacency matrix, so1 to n. So, so basically what we have? So, either we have a i j is basically is adjacency matrix means the weight of the. So, a i j. So, this is the i’th vertex, this is the j’th vertex. So, this is the a i j. So, if from i to j, v i to v j if there is edge weight and the weight of that is say w i j. So, that is basically a i j. So, a i j is basically w i j, if i and j is an edge. Otherwise it must be infinity because we cannot put 0 because 0 could be the weight of the edge. If v i j not E. So, this is the way we defined the adjacency matrix of this on this directed graph. So, this is the adjacency matrix..

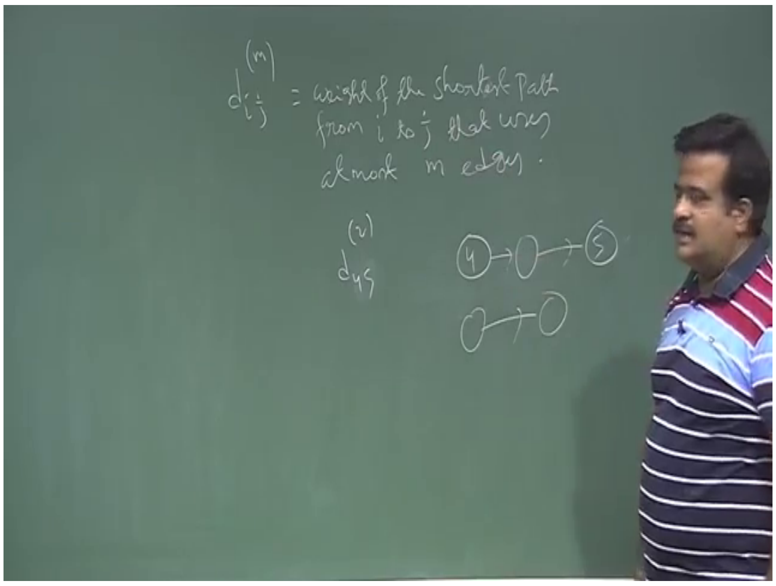
(Refer Slide Time: 13:33)


$d_{ij}^{(m)}$ = weight of the shortest path from i to j that uses atmost m edges.
$i \to \circ \to \circ \to \cdots \to j$
$\le m$

Now, we defined a d matrix basically d i j. So, we defined a d matrix d i j m, the i j’th element of this d. So, this is telling us this is the weight of the shortest path from i to j that uses at most n number of edges. That uses at most m number of edges. So, that is our d i j. So, d i j m is basically, so we take a i‘th node, this is the j’th node. So, we consider all the path from i to j like this and we count the number of edges this is 1 edge, 2 edge weight like this, we count the number of edges. So, number edges must be less that equal to m.

So, we count number of edges. So, we take the all the such path m is a variable, where m is a value m could be 1, 2, 3, 4, 5, 6 we will come to that. So, we consider all the path where in the middle in the path we are counting the number of edges. So, number of edges must be at most m. So, that is we are allowing. So, among all such path we take the path which is having the least weight. So, that is the weight of that shortest path from i to j that uses at most m number of edges.

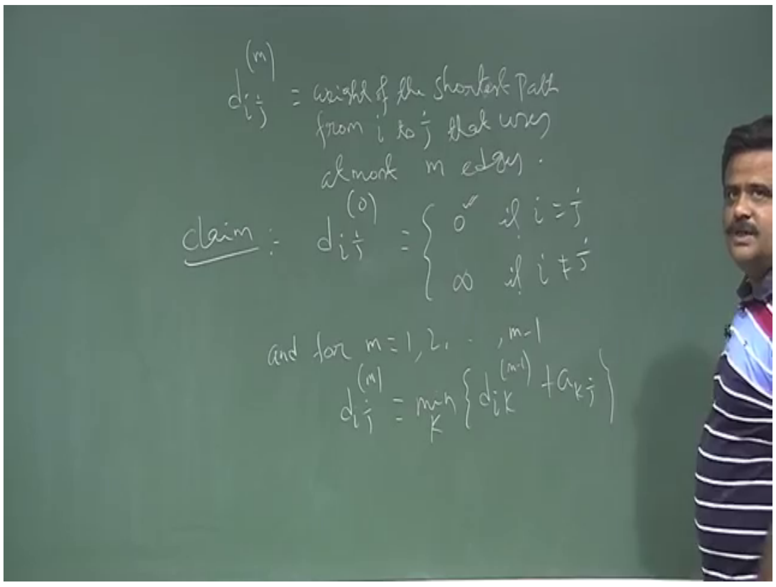
(Refer Slide Time: 15:26)


$d_{ij}^{(m)}$ = weight of the shortest path from i to j that uses atmost m edges.
$d_{45}^{(2)}$
4
5

So that means, what is say suppose we have trying to find w i j 2, say w 4, 5, 2. So, we have v 4, v 5. So, we consider all path from v 4 to v 5 in which we just take maximum two edges. So, this is one possibilities, this is one possibilities, maximum two edges. So, there may be another path which is having. So, either two edges or one edges.

So, among this paths we take the shortest path which is the least weight and that is the weight of that path is d i j m. Now we will find some expression for this, we need to have the, so in order to apply the dynamic programming we need to have some recurrence relationship.

(Refer Slide Time: 16:18)



$d_{ij}^{(m)}$ = weight of the shortest path from i to j that uses atmost m edges.

claim: $d_{ij}^{(0)} = \begin{cases} 0 & \text{if } i = j \\ \infty & \text{if } i \neq j \end{cases}$

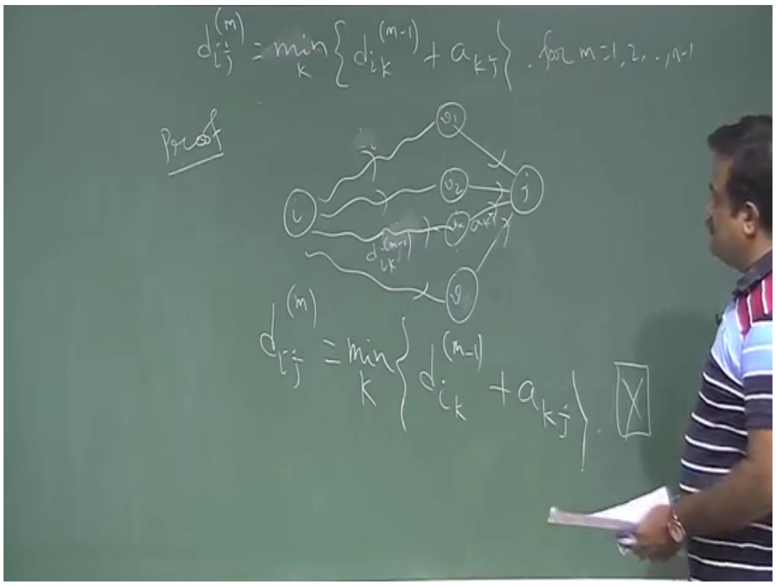
and for m = 1, 2, . . , m-1

$$d_{ij}^{(m)} = \min_{k} \left\{ d_{ik}^{(m-1)} + a_{kj} \right\}$$

So, our claim is. So, the base case, so d i j 0 is what? d i j 0 is basically 0, if i is equal to j otherwise it is infinity if i not equal to j. So, what do you mean by d i j 0? We are at i, we want to go to j and 0 means at most m edges. So, 0 means no edges we must visit. So, there is no possibilities of direct path and if i is equal to j also then we are not allowed to a cell path because that is one path. So, better thing if we remain at i. So, that is basically 0. So, that is basically i and there is no way we can go from there is no path will exists from i to j without visiting any edge because if i is not equal to j. If i is not equal to j there has to be edge to visit i to j, but we are not allowing that. So that means, there will be no much path will exist and for i is equal to j will remain at i. So, that is 0. So, this is ok.

Now, we want to have a, this is for m is equal to 0 and for m is equal to 1 up to m minus 1, we claim that this you have to prove; d i j m is basically minimum over k d i k m minus 1 plus a k j. So, this you have to prove; d i j, d i k. So, this is the general expression this is the recurrence. So, this is the. So, we have to show this. So, let us try to prove this. So, this basic is historical now this we have to prove.

(Refer Slide Time: 18:32)


$d_{ij}^{(m)} = \min_k \{ d_{ik}^{(m-1)} + a_{kj} \}$. for m=1,2,..,n-1
Proof
v_1
v_2
v_k
v_l
i
j
a_{kj}
$d_{ik}^{(m-1)}$
$d_{ij}^{(m)} = \min_k \{ d_{ik}^{(m-1)} + a_{kj} \}$.

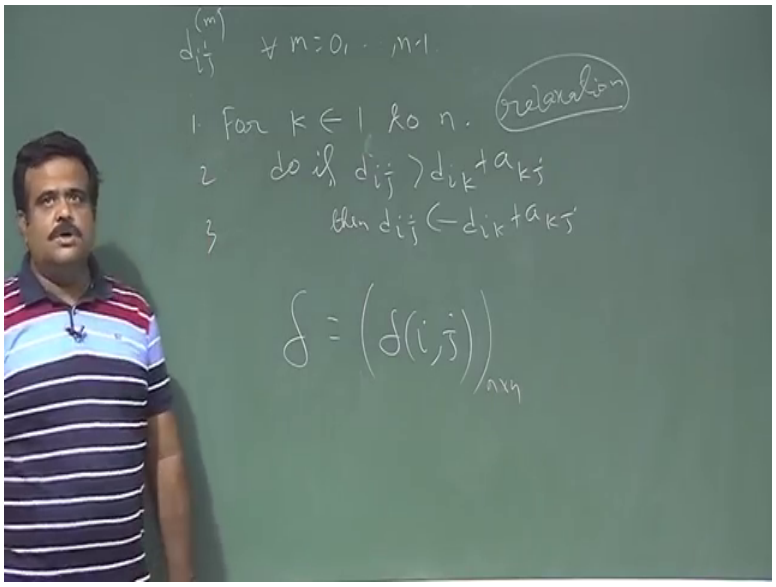
So, let us write this d i j m is basically we are maximum by sorry minimum shortest path, now minimum among all this m minus 1 plus a k j. So, this is for m is equal to 1 to up to m minus 1 sorry m minus1.

So, how to prove this? So, to prove this, we have to draw the picture. So, we are i‘th node and we want to go to j’th node. So, in the middle, so we take all the incoming edges from any other node to j’th node. So, suppose this is say v 1 another node v 2, I mean v 1, v 2 like this, so v i k. So, this are all the edges which are coming to j. These are all possible edges we are say coming to j these are direct edge. So, means suppose there are few more edges like this. So now, this is say v k is over here, so v k dot, dot, dot; so directed.

Now, we have to go to i to j with total number of edges at most m. So, we already have a edge weight so; that means, from here to here we are allowed to go by, how many edges we can see here? m minus 1 edge so; that means, this is d i. So, this is the k’th vertex. So, this is basically d i k m minus 1 sorry d i k m minus 1. So that means, from here to here we are allowed m minus 1 vertex edges to see in the path then, we can take this direct edge. So, that is basically a k j this is a k j. So, the weight total weight is this plus this we choose the minimum among all such thing. So, that is the proof. So, this is the proof. So, this means d i j m will be the minimum among all such path minimum among all such k minimum among that we go from v i k with the at most m edges then, we go from then we take the direct edge. So, this is the proof of this theorem.

So, this will give us a algorithm. So, this sort of, so this will give us a algorithm. So, now, the question is if we get all the d i j’s then, what is the benefit we are getting? So, if we get all the d i j’s, d i j m for all m.

(Refer Slide Time: 22:16)



$d_{ij}^{(m)}$ $\forall m = 0, \ldots, n-1$

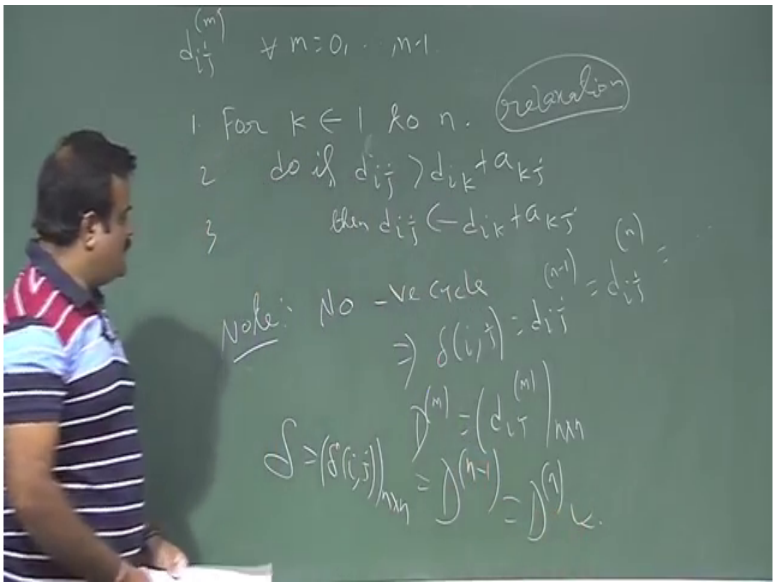
```
1. For $k \leftarrow 1$ to n.        relaxation
2.     do if $d_{ij} > d_{ik} + a_{kj}$
3.         then $d_{ij} \leftarrow d_{ik} + a_{kj}$
```

$$\delta = \left(\delta(i,j)\right)_{n \times n}$$

Suppose we are having a algorithm where we can we are just following this algorithm. So, what is the algorithm? Algorithm is like this for k is equal 1 to m because we know the zero value do if, if d i j, I mean d i j is less than is greater than d i k plus a k j. So, this should be m or yeah, but any way we are not putting the index then, we must change this d i j, d i j to be d i k plus a k j. So, this is sort of relaxation, if you recall the relaxation, this is sort of relaxation step a k j. So, this is sort of relaxation we are doing.

Now, suppose we have this, now what is the if we have this d i j's then what is the benefit we are getting? How this will help us to have a deltas? Because ultimately we are going to find out deltas no. Delta of i j this is the shortest path. So, we need to find the delta matrix basically, so all pair shortest path. So, what is the relationship between deltas and d i j’s? So, suppose there is no negative cycle; that means, there will be the shortest path delta will exists. Now what is the delta with d i j then? If there is no negative cycle then how many edges will be there in the middle? Because there all the path will be the shortest path will be the simple path then. So, simple path means there will be at most n minus 1 edges in the middle.

(Refer Slide Time: 24:20)



$d_{ij}^{(m)}$ $\forall m = 0, \dots, n-1$

```
1. For k ∈ 1 to n.          relaxation
2.    do if $d_{ij} > d_{ik} + a_{kj}$
3.       then $d_{ij} \leftarrow d_{ik} + a_{kj}$
```

Note: No -ve cycle $= d_{ij}^{(n-1)} = d_{ij}^{(n)} =$

$\Rightarrow \delta(i,j) = d_{ij}$

$D^{(m)} = \left(d_{ij}^{(m)}\right)_{n \times n}$

$\delta = \left(\delta(i,j)\right)_{n \times n} = D^{(n-1)} = D^{(n)}$ ✓.

So that means, this is a node no negative cycle implies delta of i j will be d i j m minus 1 which is basically converging d i j n dot, dot, dot, dot, dot, d i j n plus 1.

So, basically this is the deltas of i comma j and this is what we are going to find out. So, if you can find out d i j n then we have proved. So, if we can find out d i j n then we have through. So, that is basically we are that we can get by this algorithm and which is taking how many time, it is a basically it is taking. So, will take this d i j in a matrix form basically define the D matrix d of m is basically d i j of n, n cross n and. So, basically delta; delta is basically this matrix big delta; delta i j. So, this is basically our D of n minus 1 which is same as D of n. So now, in the next class we will talk about how we can get this D matrix. So, we will us the matrix multiplication technique there. So, we will discuss this in the next class.

Thank you.