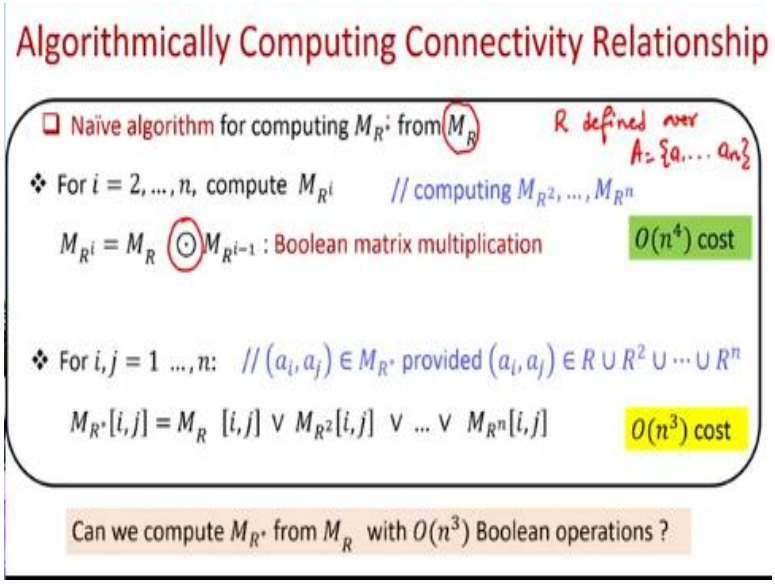
Discrete Mathematics
Prof. Ashish Choudhury
Department of Mathematics and Statistics
International Institute of Information Technology, Bangalore

Lecture -19
Warshall's Algorithm for Computing Transitive Closure

Hello everyone. Welcome to this lecture. In this lecture we will continue our discussion regarding how to compute the transitive closure. So, we will discuss the efficient algorithm given by Warshall for doing the same.

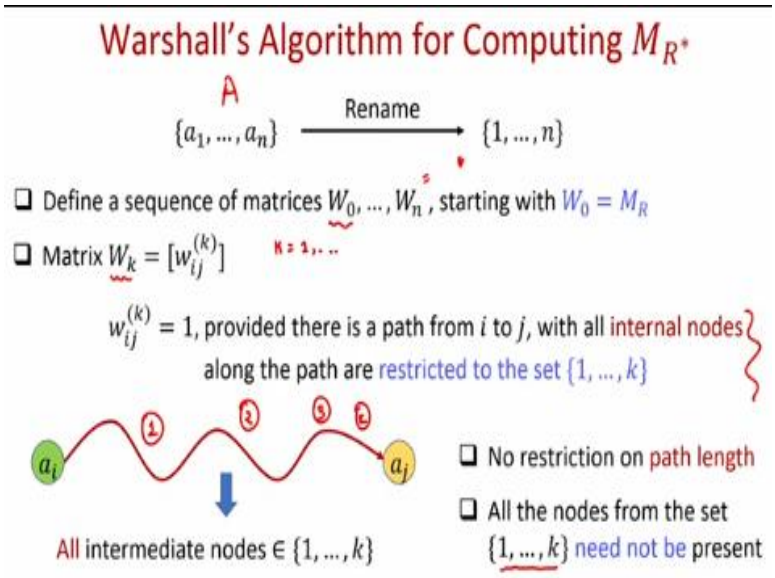
(Refer Slide Time: 00:37)



So, just to recap this was your naive algorithm for computing the connectivity relation. So, you are given as an input the matrix for your original relation R (M_R) and from that we constructed the matrix for the relation R* (M_{R^*}) provided R is defined over a set consisting of n elements. So, R defined over set A consisting of n elements. We introduced the Boolean matrix operation in the last lecture.

And we saw that overall to compute the matrix for different powers of R it will cost you n^4 Boolean operations. So, now our goal is how we can get an algorithm to compute the matrix for the connectivity relationship with n^3 Boolean operations.

(Refer Slide Time: 01:33)



So, here is the Warshall's algorithm for doing the same. So, the first thing that we are going to do is for simplicity we are going to rename the elements of the set A from a_1 to a_n as 1 to n. So, even though the set is for instance might be a set of n cities we will be attaching the labels 1 to n to those n cities. So, we will be now instead of calling a_1 to a_n we will be now calling the elements of A as 1 to n. This is just for simplicity and understanding.

Now the idea behind the Warshall's algorithm is that we are going to define a sequence of matrices where the matrix W_0 is the initial matrix namely the matrix for the relation R (M_R) given to you. And from that we are going to construct a sequence of matrices, W_1, W_2 up to W_n and we are going to stop with W_n and W_n will be the matrix for your relation R*. So, let us see the definition of the kth matrix in this sequence.

So, k here ranges from 1 to n. So, the i, jth entry of this kth matrix I am denoting by this notation ($w_{i,j}^{(k)}$) . So, you have w i, j or i j, and in the superscript I have is k within the parenthesis. So, I will have the entry number i, j in different W matrices. So, the version of the W matrix that I am focusing on will be k. So, that is why k will be superscript and within that kth version of the W matrix i, jth entry will be denoted by this subscript indices i and j.

So, the i, jth entry in matrix W_k, $w_{i,j}^{(k)}$ will be 1 will be defined to be 1 provided the following holds.

If there is a path of some length, I stress the length is not important. If there is a path of some length from node i to node j in your original graph such that all internal nodes along the path are within the set 1 to k. Now what do we mean by internal nodes? By internal nodes I mean the intermediate nodes in that path.

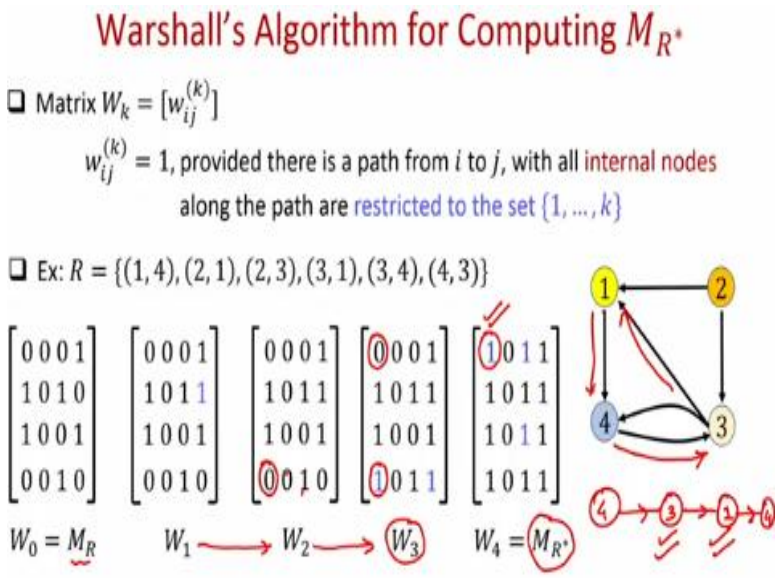
If there are no intermediate nodes that is fine, I may have a direct edge from the node i to node j. That is fine. The condition here is that if at all there are any internal nodes along the path from the node i to j they should be within the set 1 to k. That means only the nodes 1, 2, 3 or k are allowed as intermediate nodes in that path from the node i to j. If such a path is there I will say that the i, jth entry in the matrix W_k will be 1 otherwise 0.

So, pictorially, if you have such a path like this, where all intermediate nodes are restricted within the set 1 to k, I will say the entry is 1 in the i, jth entry W_k matrix is 1. So, couple of common confusions which has come to the student when we give this definition. First one is as I said here there is no restriction on this path length. This path could be of length 1, that means it could be a direct edge in which case it is not violating the condition.

Because the conditions says that if at all the intermediate nodes are there in this path they can be only node number 1 or node number 2 or node number 3 or node number k. That means the second restriction here is that it is not necessary that the path should be of length k. That means it is not required that you should have node number 1 somewhere node number 2 somewhere node number 3 somewhere node number k somewhere and then you come to node j.

No, that is not the interpretation of this definition. The definition is a conditional requirement or requirement is that if at all intermediate nodes are there, first of all there can be any number of intermediate nodes. Of course they cannot be more than k intermediate nodes because the only intermediate nodes which are allowed in the path which you are considering in the matrix W_k are the nodes now the nodes 1, 2 up to k.

(Refer Slide Time: 06:31)



That is the definition of the matrix W_k. So, before proceeding to the Warshall's algorithm, let me give you an example to make clear what exactly I mean by these W_k matrices. So, imagine this is your given relation R. So, the diagram or the directed graph for this relation is as follows. So, the nodes will be 1, 2, 3, 4. Since 1 is related to 4 you have this directed edge, 2 is related to 1 you have this directed edge and so on.

We start with the matrix W_0, which is the matrix for your relation R (M_R) . So, what I have done here is that I have just added the entry 1 for (i, j) pair if i is an edge to the node j otherwise the entry is 0. Now let us see the matrix W_1 here. How do we compute matrix W_1? Forget about it. The interpretation of W_1 matrix is that the i, jth entry will be 1 if a path with only node 1 as intermediate node is present from i to j.

So, it turns out that if I say for instance the first three columns for the first row then they will be 0, 0, 0. Why so? Because; there is no path from the node 1 to the node 1 where only intermediate nodes are the node number 1. So, you do have a path here from the node 1 to node 1. So, you can go via 1 to 4 and then you can go from 4 to 3. And then you can go from 3 to 1.

That is a path from 1 to 1. But what are the intermediate nodes here? So, you started with 1, you go to 4, you go to 3 and then you go to 1. So, well there is a path from node 1 to 1. But what are the intermediate nodes? The intermediate nodes here are node number 3, node number 4. But

they are not allowed to be considered when you are considering the paths in matrix W_1. This will be considered as a valid path when the possible intermediate nodes are nodes 1, 2, 3, 4.

Namely when you are considering going to consider the matrix number W_4, this will be considered as a valid path. But right now when you are considering paths with respect to the matrix number W_1, this is not a valid path, this is not allowed here. This is not why 1, 1 is 0 here. In this, due to the same reason if I consider say for instance the entry number 2, 2 it is 0. Because even though I have a path from 2 to 2, you can go from 2 to 1, you can go from 1 to 4, you can go from 4 to 3. Sorry here you have no path, sorry. So, the entry number 2, 2 anyhow will be 0.

So, let us see a path which is not allowed as a valid path in matrix number W_1. So, if I consider let me show why this entry 2, 4 is becoming 1 here. So, the entry 2, 4 was 0 in matrix 0, W_0. Because there is no direct edge from 2 to 4, but now since I am allowing node 1 to be included as an intermediate node, I can see that there is a path from node number 2 to node number 4 going through this intermediate node 1.

That means once I allowed node number 1 as possible intermediate node, I can get a valid path and that is why node is entry 2, 4 becomes 1 here and then you can verify that other entries are 0, will remain 0 here. Now let us see whether there exist, whether I will get any update in matrix W_2. That means now in W_2 I am allowing you to include node number 1 and node number 2 as intermediate nodes between any a_i and any a_j.

It turns out that if I include node number 1 and node number 2 as possible intermediate nodes the status of any i, j pair does not change when I go from W_1 to W_2. But when, I go from W_2 to W_3, then the status of this entry 4, 1 changes. So, the 4, 1 entry was 0 in the previous matrix, but now it is becoming 1 when I am allowing you to include node number 3 as well as a possible intermediate node and why so?

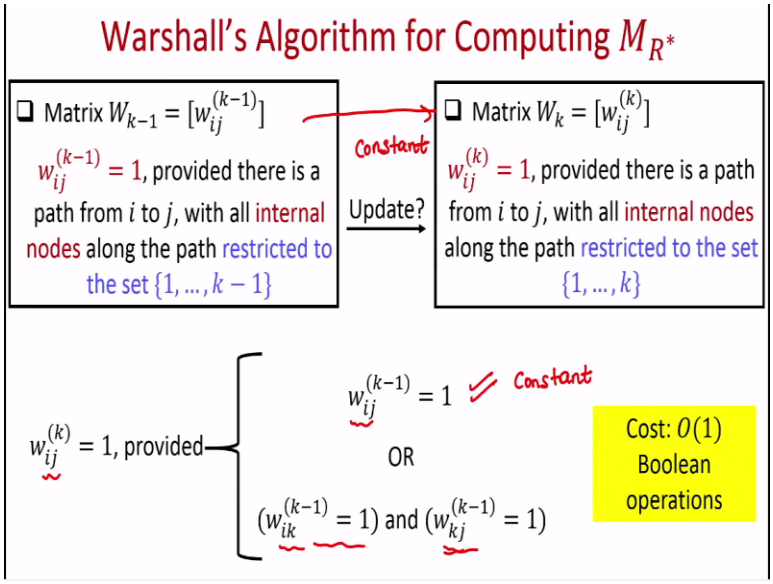
Because you see here that I do have a path from the node number 4 to node number 1. The path is as follows, you go from 4 to 3, you go from 3 to 1 then you go from 1 to 4 and what are the intermediate nodes here? The intermediate nodes are 3, 1. And, 3 and 1 are allowed to be

intermediate nodes when you are considering matrix W_3. See the idea here is that as I keep on going to the next sequence of matrix and I am allowing you to include more and more new intermediate nodes as valid intermediate nodes in the path from i to j.

So, in matrix W_2 I was not allowing you to include node number 3. I was only allowing you to include node number 1 and node number 2. But when I go to matrix W_3 I will be giving you more flexibility. I am giving you the flexibility to even include node number 3 along with node number 1 and node number 2 as possible intermediate nodes. Now you can check that when I update W_3 to W_4, the entry 1, 1 becomes 1 which was 0 in the previous matrix.

That means as long as I allow you only nodes 1, 2 and 3 as intermediate nodes I will say there is no valid path from 1 to 1 going through only these three intermediate nodes. But as soon as I allow you to include node 4 as a possible intermediate node I will say that there exist a valid path from 1 to 1 and now I can stop here with W_4. That will be my matrix for R* and this is because you can check that only those nodes which are reachable from any other node for those particular entries the i, jth entry will become 1 here in W_4. Whereas the 0 entries means that there exists no path from the node i to node j.

(Refer Slide Time: 14:34)



So, now Warshall's algorithm boils down to the following that how exactly I am going to update my matrix W_{k-1} to the matrix W_k and as per my definition the i, jth entry in the matrix, W_{k-1} is 1

provided I have a valid path satisfying the restriction that the intermediate nodes are within the subset 1 to k – 1 and I want, assuming I have already computed matrix W_{k-1}, my goal is how to find out the matrix W_k, where, I am allowing you to include node number k as well, as a possible intermediate node.

So, the update is very clever. It is done as follows. There can be two possible cases. If your i, jth entry in the matrix W_{k-1} is 1 then I can simply say that i, jth entry in matrix W_k will also be 1 and this is because any valid path from node i to j where the intermediate nodes are within the subset 1 to k - 1 can also be considered as a valid path, where the intermediate nodes are restricted within the subset 1 to k.

And this is because as per the definition of matrix W_k valid paths are not supposed to have all the nodes in the set 1 to k. The definition says that the path will be valid if at all the intermediate nodes are within the subset 1 to k. It will not have all the nodes in the set 1 to k. So, it is fine if I do not include node k in a path from i to j. If such a path excluding node k is also already present and known to exist in the previous matrix W_{k-1}, I can say that that is still a valid path in the matrix W_k.

And I can check whether the i, jth entry in the matrix W_{k-1} is 0 or 1. If it is 1, I can simply copy that entry in the new matrix. The second case will be as follows. My claim here is that if in the (k – 1)th matrix the i, jth entry is 0. That means there was no valid path from the node i to node j where all intermediate nodes are restricted to the subset 1 to k - 1. No such path was there. Then what I will check is the following.

I will check whether there exists a path from the node i to k where all the intermediate nodes are within the subset 1 to k - 1. So, that will be done by checking whether the i, kth entry in the (k – 1)th matrix is 1 or not. That means, I am checking whether there exists a valid path from the node i to k in the k – 1th matrix. And I will check whether there exists a valid path from the node k to the node j in the k - 1th matrix.

Now what I can say is if these two paths individually are guaranteed to exist then if I combine or

if I merge these two paths then that will be considered as a valid path from the node i to node j passing through all the intermediate nodes that were there in the previous paths plus a new intermediate node namely node number k and this will be considered as a valid path for W_k matrix.

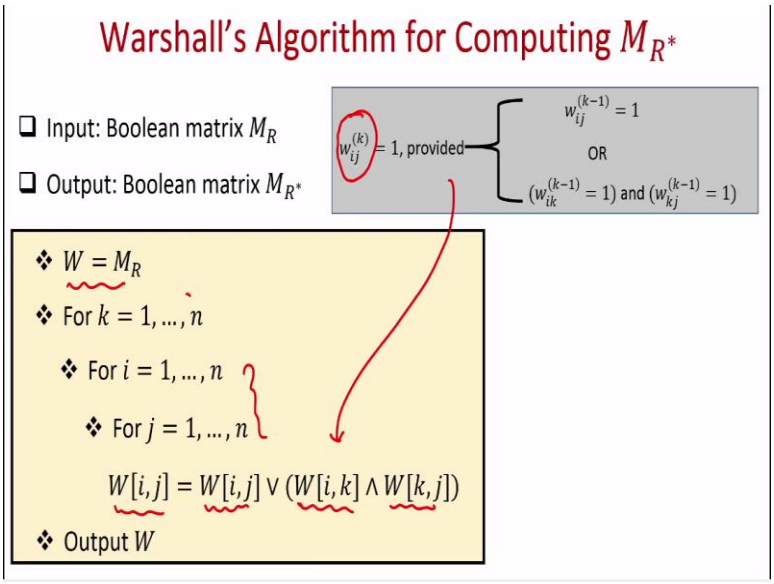
I stress this will not be considered as a valid path for the W_{k-1} matrix because in W_{k-1} matrix node k is not allowed as a possible intermediate node. Node k can be allowed as a valid intermediate node only in the paths considered in the matrix W_k. So, intuitively what I am saying is that in the absence of intermediate node k there was no path from the node i to node j. But as soon as I include the possibility of having node k as a valid intermediate node I can say that a path from i to j is there, provided there is a path from i to k and individually a path from k to j which is equivalent to checking the conjunction of these two conditions. I have to check whether the entry number i, k and the previous matrix is 1 or not and whether the entry k, j is 1 or not in the previous matrix. If both of them are 1, I can conclude that the entry i, j in the new matrix will be 1. That is the intuition for the update in case two and that is a clever intuition.

So, what I can say is that I can summarize the two cases for updating the i, jth entry from the previous matrix to the new matrix as follows. If in the previous matrix i, jth entry is already 1, I will say that even in the new matrix the i, jth entry will be 1. Else I will check whether the i, kth entry as well as the k, jth entry are simultaneously 1 in the previous matrix and if that is the case I can say that i, jth entry in the updated matrix is also 1.

And what will be the cost for performing these operations? So, this will need constant amount of effort because you just need to go and check the i, jth entry of the previous matrix and in the same way you can go and just check the i, kth entry of the previous matrix and the k, jth entry of the previous matrix we just need to do two matrix lookup, which will be costing you cost amount of effort. That means to do this update operation you just need to perform a constant amount of operations. And how many such i, j pairs are there?

There are n^2 i, j entries, for each i, j update you have to spend a constant amount of effort O(1) and that is why for n^2 entries the overall update cost will be n^2.

(Refer Slide Time: 21:48)



So, this is different from the way we performed update in the naive algorithm. In the naive algorithm we were naively computing the higher powers of R from the matrix of the previous power of R. And each update was costing us n^3. But here we are not doing a naive update over matrices, W_0, W_1, W_k, W_n have different interpretations. And due to the different interpretation the update from W_{k-1} to W_k will be now costing us only n^2 effort instead of n^3 effort.

So, now let us put together everything and get the pseudo code for Warshall's algorithm. I am retaining the update operation of the Warshall's algorithm. How do you update update the k - 1th matrix to get the kth matrix? So, the input here will be the matrix for your original relation R and now what I will do is I will run 3 loops each ranging from 1 to n and the update operation that I am going to perform is as follows.

I am just copying the update operation as it is, removing all the subscripts and superscript notation. So, W[i, j] denotes the updated i, jth entry and how it is updated? I check whether the previous i, jth entry is 1 or not or if the i, kth entry as well as the k, jth entry are simultaneously 1 in the previous matrix or not. If that is such case I will update W[i, j] from 0 to 1. Otherwise I will set W[i, j] to 0 and this I have to do for k = 1 to n.

So, it takes care of the fact that I am doing the update operation n times and for each time I am

doing the update for each of the n^2 entries of the form i, j. And then finally I will output the W matrix which will be the matrix for my connectivity relation. And it is easy to see that this overall operation algorithm will cost you only $O(n^3)$ effort. Because there are three loops each running from 1 to n.

So, that brings me to the end of this lecture. Just to summarize in this lecture we saw the Warshall's algorithm for computing the connectivity relationship which is better than our naive algorithm for computing connectivity relationship. Thank you.