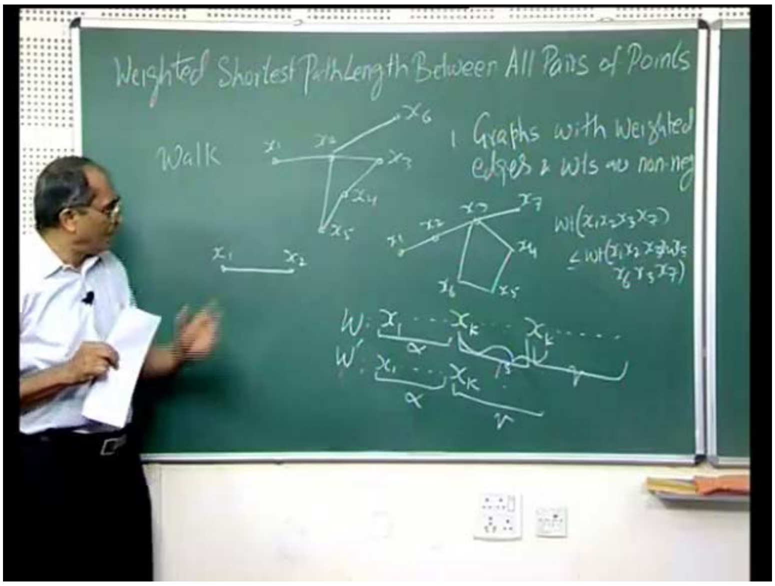
Computer Algorithms - 2
Prof. Dr. Shashank K. Mehta
Department of Computer Science and Engineering
Indian Institute of Technology, Kanpur

Lecture - 4
All Pair Shortest Path

So, today we will discuss an algorithm to compute the weighted shortest path length between all pairs of points. Earlier we have discussed Dijkstra algorithm, which was to compute the same quantity where one of the n point for all paths was fixed, namely s. So, one can compute this by running Dijkstra algorithm, for each value of s ones so running it n times. So, today we are going to discuss another algorithm. Now, before we begin with this problem, I would like to discuss the same problem, but instead of paths we focus on walks.

(Refer Slide Time: 01:07)



Let me remind you a walk is a sequence of vertices. So, say this is x 1, x 2, x 3, x 4, x 5, again x 2, x 6. The sequence of vertices such that there is an edge between every successive vertex; so it is possible that you may revisit a vertex or an edge more than one while in a path you never revisit a vertex. So, it is clear that there are several more walks than paths in a graph, in a graph with finite number of vertices number of paths can be finite. Cannot be infinite, because there are only n factorial possible sequences of n vertices, so one can have at most i factorial possible sequences i equal to 1 to n.

Some of them are the paths and others are not even a paths, but in terms of walks this number is in general infinite. So, for example- if I have a simple graph with only two vertices and an edge so I can have a walk, which goes from x 1 to x 2 and then back to x 1 and again to x 2 and so on. So this can go on forever, so the number of walks are infinite and the walks themselves can be unbounded in size. So, if I want to look at all possible walks. Then determine the weighted shortest walk, then what is the relation between the two quantities?

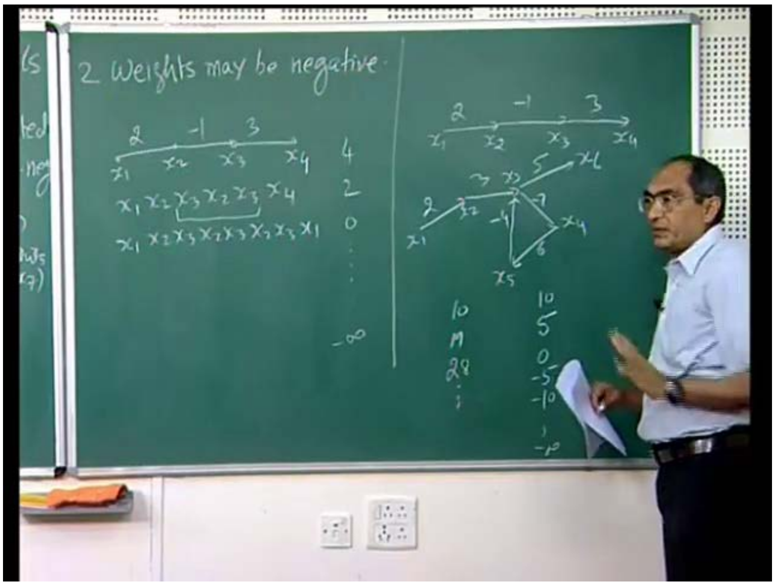
So, let us first talk about graphs with weighted edges and weights are non negative, so first we consider only the non negative weights. Well in that case, suppose a certain walk x 1, x 2, x 3, x 4, x 5, x 6, x 3 and then x 7. Suppose, we have a walk which goes like this revisits x3 comes to x7. Now, because the weights are non negative, then let us compare the weight of this walk with the path x 1, x 2, x 3, x 7.

So, clearly the weight of the path x 1, x 2, x 3, x 7 is going to be less than equal to weight of x 1, x 2, x 3, x 4, x 5, x 6, x 3, x 7, so this suggests that, probably a shorted shortest weighted walk is likely to be a path. This is true because whenever a walk revisits a vertex so suppose a walk w starts at some x 1 goes to some x k and revisits x k. Then goes on then you have a cycle here and weight of the cycle is non-negative.

So, we can replace this by another walk w prime x 1 which is say x k so this is say alpha, this is beta, and this is gamma then we can write down x 1 x k and so on where this is alpha and this is gamma. So, I can cut off the cycle and I get this the weight of this w prime is less than equal to weight of w, and I can go on reducing and eventually it will become a path. So, I can always find a path with weight less than or equal to this it is possible let a walk, which is not a path also have a minimum weight and that will happen if the cycle involved, or cycles involved has zero weight apart from that it will always be a path.

Even if you compute such a walk in your computation you can always remove cycles from that and get a path out of it. So, what this suggest is that in case the weights are non-negative, then computing the weighted shortest walk length between a pair of points, is the same problem as computing weighted shortest path length between a pair of points. So, we can take advantage of this observation and replace this problem by the problem where we enlarge the search space and put walk here.

(Refer Slide Time: 07:52)



Now, let us talk about the case second case where the weights are weights may be negative. What, happens in this case when we are searching the minimum weight walk in this case suppose, let us take a simplest simple example suppose there is a weight 2 and a minus 1 and 3, and this is x 1, x 2, x 3, x 4. The, walk from x 1 to x 4 can be x 1, x 2, x 3, x 4, which is the simple path that, we have the another walk could be x 1, x 2, x 3, x 2, x 3, x 4, which involves this cycle. Note that this, I am going back its a closed walk I can go on doing this I can have x 1, x 2, x 3, x 2, x 3, x 2, x 3, x 1.

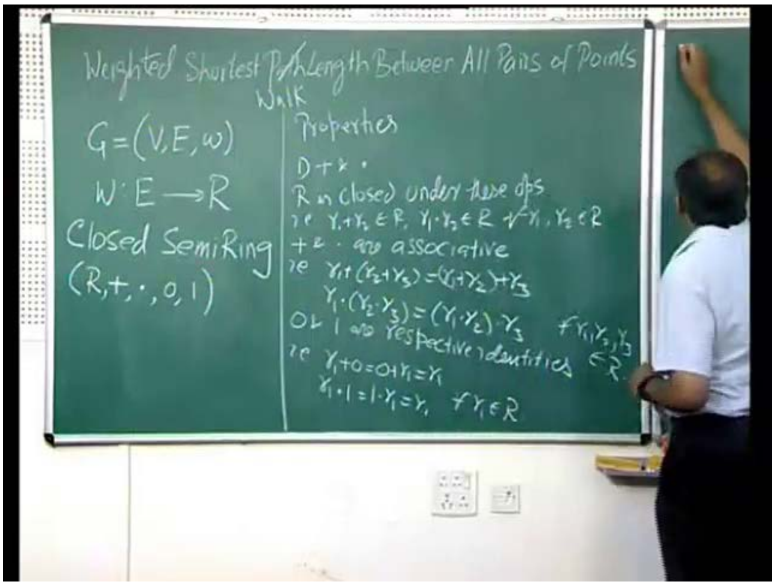
Note that the weight of the simple path is 4, over here it is 2, minus 1, minus 1, minus 1 and 3, so it is just comes down to 2 this is 0. Now, we will keep on decreasing in fact it will become unbounded will go to minus infinity. So, the shortest walk is not even defined in this case, even when there is one edge with a negative weight. So, clearly the problem cannot be solved for the shortest path by replacing them by walks. But what happens when you have directed graphs, so let us take case 2 minus 1, 3.

Well, the same problem is not going to arise, because x 1, x 2, x 3, x 4, x 1, x 2, x 3, x 4 is a valid walk as well as path, but x 1, x 2, x 3, x 2, x 3, x 4 is no longer relevant because x 3, x 2 is not in right direction so we do not have much problem. Now, let us take another case let us say 2, 3 minus 4, 7, 6, 5 x 1, x 2, x 3, x 4, x 5, x 3, x 6 I have several walks.

Now, I can go like this or I can as well the simplest thing is just going like that x 1, x 2, x 3, x 6, or x 1, x 2, x 3, x 4, x 5, x 3, x 6 or I can keep on adding these cycles inside this, so there are infinite number of walks. But note that the weight of the successive walks increases that is because, the weight of the simplest path is 10. Then if I add the cycle, I am adding a 13 minus 4, 9 more weight so it becomes 19. If I add 2 times it becomes 28 and so on. So, in this case the shortest walk is still a path, the problem will arise when suppose the weight of the negative weight of the cycle is negative.

So, let us say we have minus 7 here in that case, the straight path the walk without cycle has weight 10, and when I go through this loop its minus so its 5 because we are adding minus 5 to it. And, you keep doing it so it will become 0 then minus 5, then minus 10 and will go to infinity minus infinity. So, the problem again arises in this case and we can no longer solve the path problem by focusing on walks, so what we notice here is that as a longnes graph is a directed I can allow negative weights, when there are no cycles of negative weight. Then, we can deal with the problem with by replacing paths by the walks.

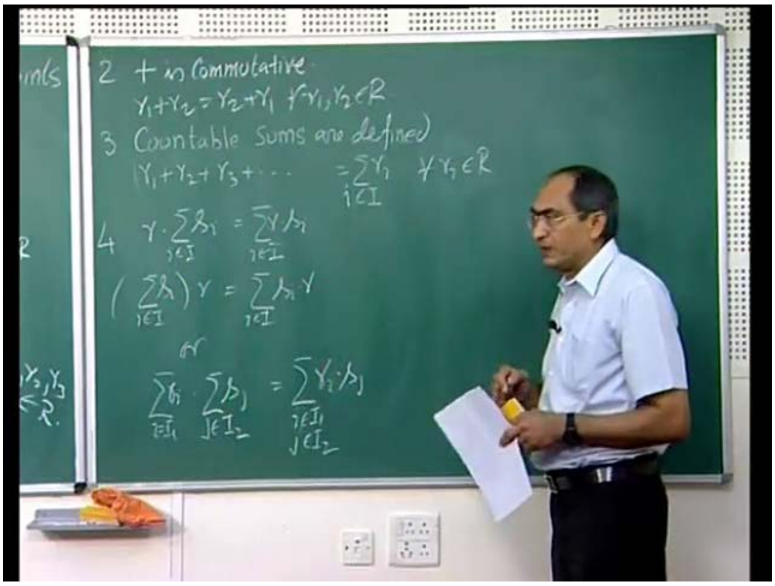
(Refer Slide Time: 13:34)



So, what we have decided is we can replace this word by walk and go on, and solve this problem. Now, this time unlike in the Dijkstra's algorithm, we are going to make one change and we will describe edge weighted graph in the same fashion, but the weight function will be a function from E to a structure which we will call closed semi ring. So, what is a closed semi ring, this generalization is interesting and it could be applicable in more than the real numbers, a closed semi ring is a couple containing these components plus and 0. This is a set of elements these two are binary operators, the plus and the product operator and these two are special elements of R.

They satisfy the following properties. That, plus and dot related properties are that R is closed under these operations, that is r 1 plus r 2 belongs to R and r 1 dot r 2 belongs to R, for all r 1 and r 2 in R. Secondly, the operation both operators are associative, dot are associative, which means that is r 1 plus r 2 plus r 3 is r 1 plus r 2 plus r 3 r 1 dot r 2 dot r 3 is r 1 dot r 2 dot r 3, for all r 1, r 2, r 3, in R. So, they are associative and 0 and 1 are respective identities, which means r 1 plus 0,0 plus r 1 is r 1, and r 1 times 1, is 1 times r 1, is r 1 for all r 1 in R.

(Refer Slide Time: 17:49)

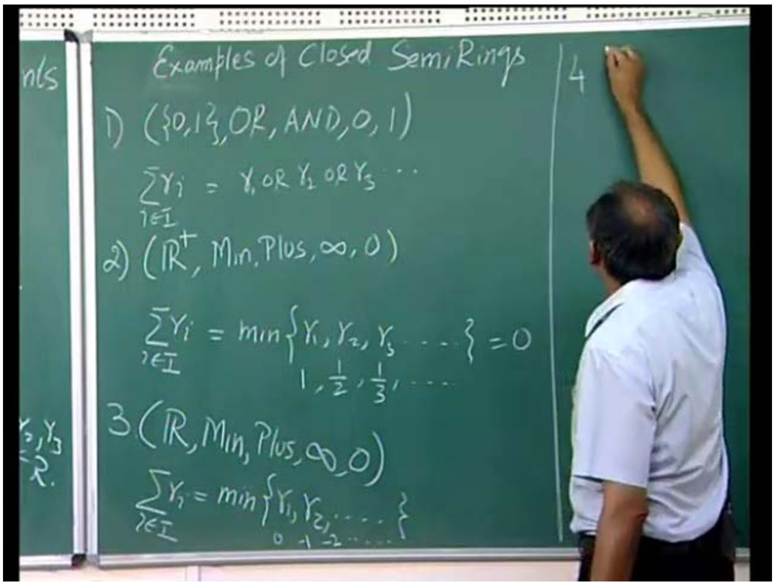


Then we also assume that plus is commutative and that means r 1 plus r 2, is r 2 plus r 1, for all r 1 and r 2 in R. Third property is that, countable sums are defined which means for any countable collection of elements of r if you take their sum is defined, which means that no matter how I associate them this sum is uniquely defined. So, we can also state this as r i, I belonging to sums index set i which is countable it could be infinite for all r i in R.

Lastly yes i belongs to capital I you are right, for all here it is, for i belongs to this and for all r i in R yes, thank you. Last is the distributive property, and which says that r times sum s i, I belongs to i is sum r times s i, and sum s i times r, is sum s i times r, i r these properties also hold. Where, this is equivalent to same or you can also say that sum r i, I in I 1 times sum s j, J in index at I j 2, i 2. Both these sets can be infinite, as long as they are countable the sum is r i, s j i, in I 1 and j in I 2, so these quantities exist. Anything that satisfies these condition will be, called closed semi ring so I am going to.

Student: Here we have not assumed that, we have not assumed that 0 is not equal to 1.

(Refer Slide Time: 21:13)

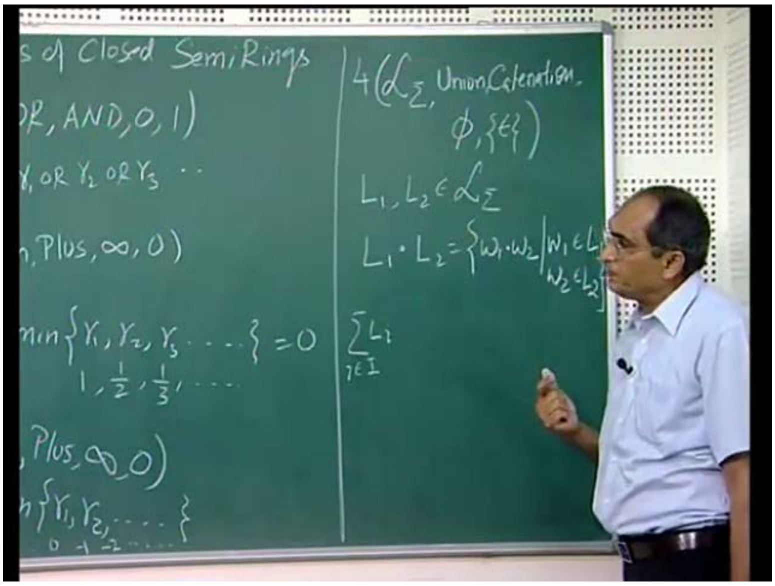


So, I will take some examples closed Semi Ring, the first one is the Boolean ring, all properties that we have described are easy to see hold in this case. The, infinite sum for example r i plus becomes r 1 or r 2 or r 3 if even 1 of these entities is 1, then this is 1 otherwise it is 0, so it is well defined other properties are easy to check. Let us take the second example, here the set is the set of non-negative real's the plus operation is minimum the product operation is the normal plus, I am assuming that plus infinity is a member of this set so plus infinity and 0.

So, in this case an infinite sum r i, is minimum of r 1, r 2, r 3. Now, a particularly nastik is of this is for example, say this is 1, this is 1 over 2, 1 over 3 you consider this does not seem to have a minimum, but we stretch this idea and say that the limiting value of this which is which will take to be 0. Although, it is not an element in this, but arbitrarily close element to this is present in this, other properties are again very easy to verify in this ring. Let me take a similar ring, but we do not restrict the numbers to non-negative real's, in this time we are allow all numbers.

Then, this also satisfies every property, but in this case if you take a look at the sum and if I take for example, 0 minus 1, minus 2 and so on then this quantity approaches minus infinity. Note that a similar concept of arbitrarily closed we cannot take the limit, because here there is always an element which is arbitrarily closed to 0, but that is not the case here so this minimum does not exist. So, this is not a closed semi ring but we will use this particular ring ensuring that, whenever we come up with infinite sum the sum exists.

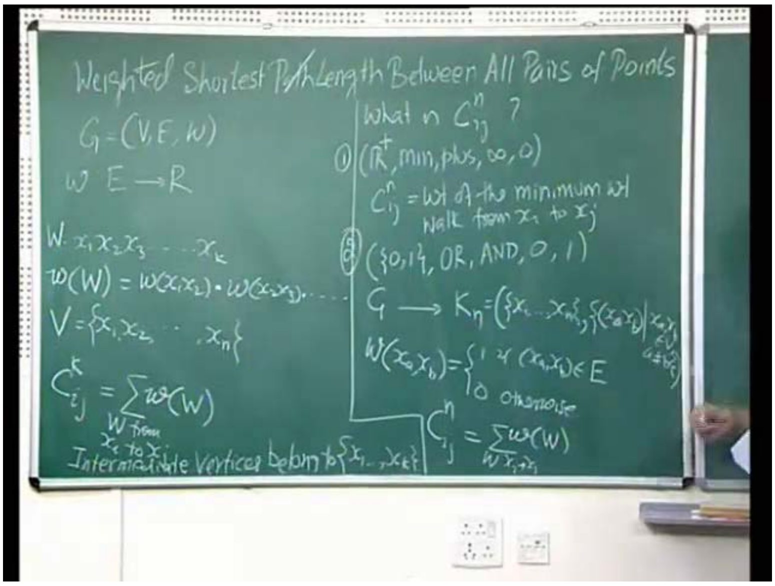
(Refer Slide Time: 25:57)



Under a fourth example, is on set of languages consider a symbol set sigma then l sigma denotes the set of all languages on sigma the plus operation is union, product operation is catenation. The, two elements special elements are the empty language, and the language containing the empty string only. Here catenation operation is as follows, given any 2 members L 1 and L 2, in l sigma. L 1 catenation L 2 is w 1 dot w 2, this is the catenation of 2 strings where w 1 is in L 1, and w 2 is in L 2.

So, with this definition of the operator catenation all again verify that, this is an example of a close semi ring. The, problem with this ring is that typically infinite sums such as say L i, any how the members of L i are generally infinite sets. Hence these are also likely to be infinite. In computation usually we do not deal with the infinite quantities. So, this will not turn out to be a useful ring for our application.

(Refer Slide Time: 28:06)



Now, we have a graph where weights have associated with the edges and the range of this weight function is a close semi ring. And, now we want to compute the weight of the shortest weight walk in the graph. So, let us because now it is no longer real number let us consider a walk. Let us suppose it is, x 1 a walk w capital W is x 1, x 2, x 3, x k which may allow multiplicity of a vertex, but what is important is there is an edge between x i and x i plus 1.

So, we define the weight of, I should use the small w the weight of the walk w is the product of the weights of various edges. This denotes an edge this is the, weight of that edge which is a member of the close semi ring and we have this operation defined and remember it is associative operation. I am not putting any brackets around it. So, this quantity is the weight of the walk given here, now we will assume that the vertex set is x 1, x 2, x n, I associate some index from one through n to the vertices of V.

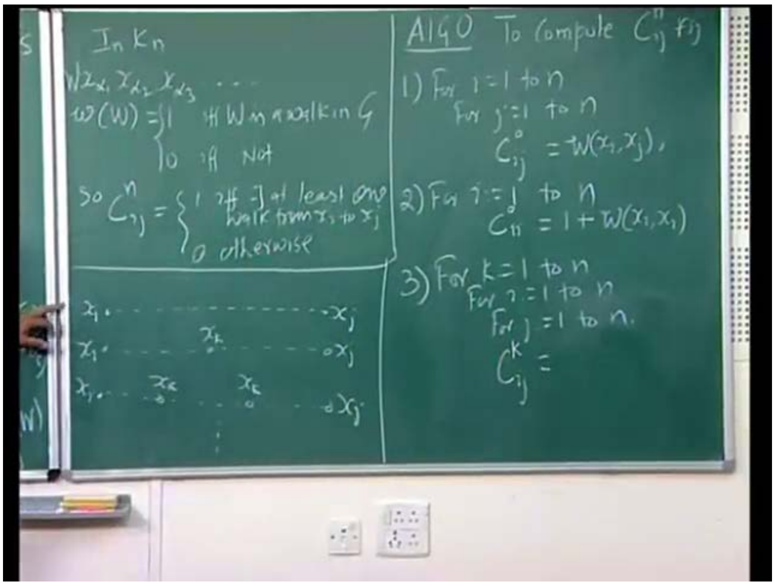
Then, let us define a quantity namely C k i j is the sum remember, there is a sum operation in the close semi ring here is the sum of the weights of the walks. And, these walks are the walk w from x i to x j, but there is one more restriction. That restriction is that except for the first and the last vertex of the walk all the intermediate vertices must belong to the set C 1 through, x 1 through x k. So, we have to add that intermediate vertices belong to x 1 through x k only.

So, we have this restriction subject to this restriction every walk qualifies here, if it starts at x i and ends at x j that sum. So, let us quickly take a look at what is this quantity for k equal to n in these examples, so What is C n i j? When, this is n the restriction is lost it has no meaning, because then it allows all possible vertices. Let us take a look at, the ring R plus, min, plus, infinity and 0, in this case we are computing the minimum of the weights of those walks which start at x i and end at x j.

So, this is C n i j is the weight of the minimum weight walk from x i to x j, this is precisely what we want to compute. This is the first example, second take a look at the Boolean range in case of the Boolean range, we had the set of Boolean constants the OR, AND, operation, AND 0, AND 1 this is what the ring walks. Now, what we have here is an a graph G to start with. But in this case what we will do is instead of working with G will work on a complete graph which is denoted by K n, which, means the graph containing same set of vertices namely x 1 through x n, but every edges allowed.

So, this is nothing but x i, x j or rather let me use some other symbol x a, x b for x a, and x b in V, n a not equal to b complete graph. We will define to capture G will define weight for x a, x b edge to be 1, if so x a, x b belongs to G to the edge set of G 0 otherwise, note that in k n every pair is an edge so we are assigning the weights in the following fashion. This essentially captures graph G, inside k n for this particular weight set the value of C n i j will be, the sum which is the sum of all the walks from x i to x j. We have weight of the walk. What is the weight of a walk in this ring? Well every sequence of vertices is a valid walk in k n.

(Refer Slide Time: 37:15)



So, in k n x alpha 1, x alpha 2, x alpha 3, is a walk but if this is the walk. What is the weight of the walk? But it is easy to see that if all of these are edges in G then and only then, this will be 1 otherwise it will be 0. So, this means if and only if w is a walk in G, if and only if not that is otherwise. So, it captures the fact that this is a walk n G when the value is 1 and when you are summing you are essentially doing OR. So, this will be one precisely there is at least one walk from x i to x j in G, so C n i j is 1 if there exist at least one walk from x i to x j 0 otherwise.

It is easy to see that this is also saying if there, if and only if there exist a path from x i to x j this is the meaning of C n i j in the Boolean ring. Now, our primary interest is of course in the first ring where we are computing the minimum, the weight of the minimum weight walk which we have seen will be same as that, the weight of the minimum weight path if we are dealing with undirected graphs. The weights are non-negative integers or it is also the same when the negative weights are allowed in directed graphs, but there are no negative weighted cycles.

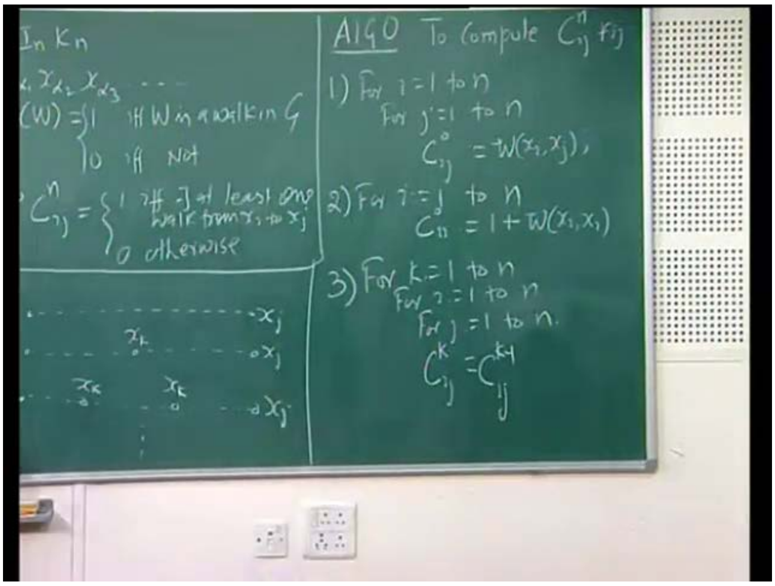
So, let us now talk about the algorithm to compute C k i j, C n i j for i, but we will compute actually all C k i j. So, the step 1 is to compute to initialize the value we will do this computation by increasing index k here, so we will begin with C 0 i j, for i equal to 1 to n, for j equal to 1 to n, C 0 i j. Well this is the sum of all the walks from i to j not allowing any vertex outside this collection this is empty set.

So, the only possible walks that can be counted here are the edges that start from x i and end up in x j. So there will be a walk, one walk to consider and that will be having the weight w x i, x j this is the only possible walk this is 1 edge walk hence this is also the sum of all the walks. And, everything is fine except when i is equal to j, in case i is equal to j we will have to separately set this, so for i equal to 1 to n. You have a vertex x i, there are two possibilities that you start here and with the empty string you stay there that is a walk, or you have a loop the loop has weight w x i, x i.

The, weight of the empty string will be 1 which is the identity of the product, remember the product is the operation we use to compute the weight of the walk. So, we will set this C 0 i i equal to 1 plus w x i, x i, now we are ready to compute the C i j for higher indices. So, will put this into another loop for k equal to 1 to n, for i equal to 1 to n, for j equal to 1 to n, and we want to compute C k i j got to return in this. Now, to compute this we have to see how this can be computed from the C values between all pairs of vertices. But the index here is up to k minus 1 because those are the values we have already computed.

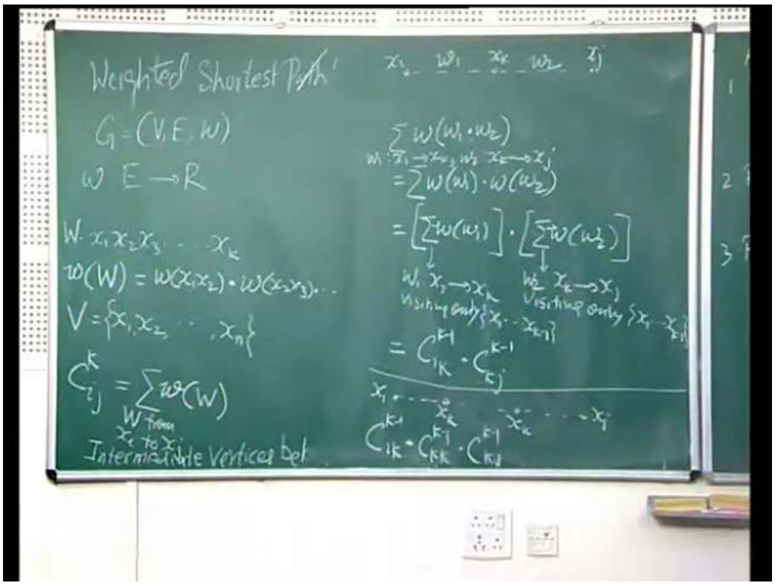
So, let us take a look at a walk from x i to x j, this is a sum of the weights of all the walks that go from i to j and the intermediate vertices are allowed to be x 1, x 2, x 3, up to x k. Let us take such walks, and classify them, partition them into several sets. First I will assume that all the intermediate vertices in this walk are from set one through k minus 1, we do not have any x k in this. Then, we will consider those, which have one occurrence of x k. Then, all sorts of walks, where there are two occurrences of x k, and so on collectively these account for all possible walks that are of any interest here. Let us find out what is the contribution of these walks to this sum, well clearly the weights of the walks that fall in this collection we add up to precisely.

(Refer Slide Time: 45:52)



C k minus 1 i j, because we never visit x k we only restrict our visits to x 1 through x k minus 1, let us take a look at the second type of walks and we take this walk and split it into 2 parts the weight of walk is weight of walk x i to x k and from x k to x j.

(Refer Slide Time: 46:36)

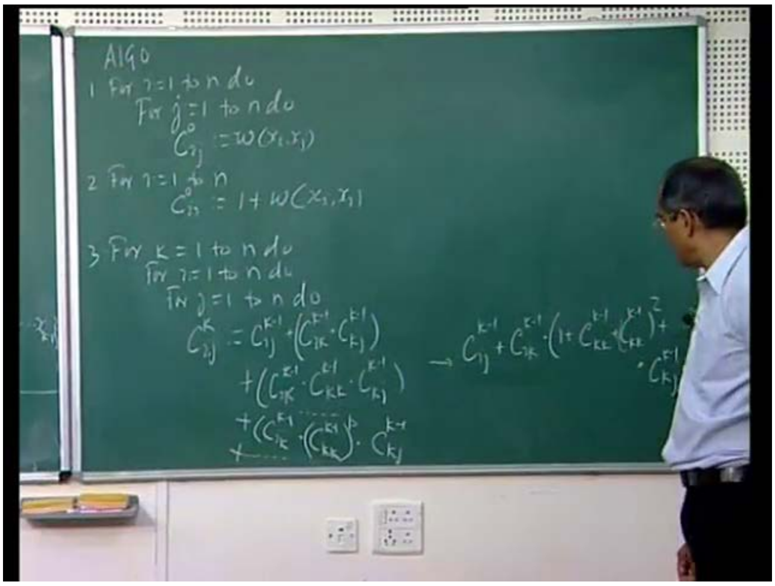


So, let us find out what is the sum of those weights all these walks start at x i, reach x k and then start at x k end at x j. Let suppose, this is sum walk w 1 and this is sum w 2. We would like to sum the weights of these walks weight of, w 1 dot w 2 this dot just indicates this is a walk, this is not an operator this is just a consumption of two walks.

But this is same as the weight of w 1 times weight of w 2, because the way weight was computed was the product of the weights of each edge, so this is the weight of this walk.

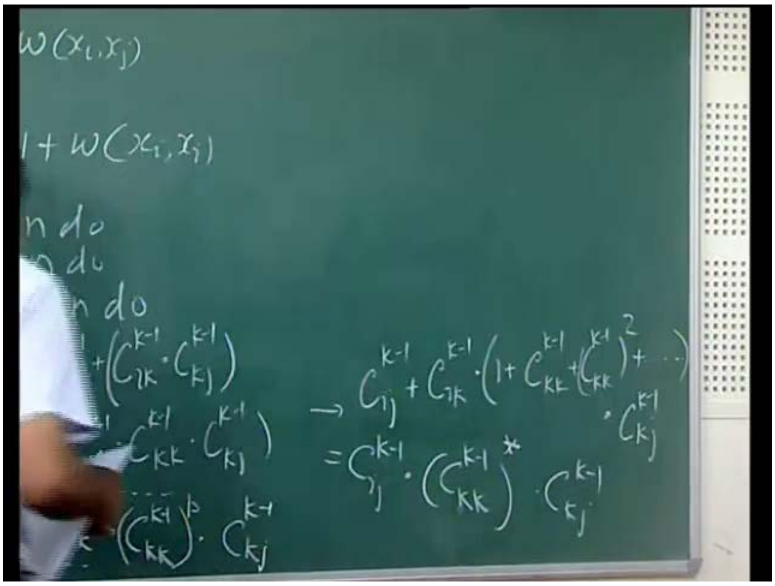
Now, we allow all possible walks from x i to x k here in all walks to x k to x j, so this can be written as sum weight of w 1 dot sum weight of w 2. Where, this is a walk w 1 from x i to x k visiting only x 1 through x k minus 1, and here w 2 is from x k to x j visiting only x 1 through x k minus 1. So, in the second type of walks what we have here is the C k minus 1 i k, dot C k minus 1 k j. Now, let us take the third type of walk. The, third type of walk start at x i go to x k then revisit x k and then get back to x j, so by the similar argument one can show that this type of walks will contribute C k minus 1 i k, dot C k minus 1 k k, dot C k minus 1 k j. This way we will have all successive terms coming in, so we can now write down the value of C k i j in terms of C k minus 1 terms.

(Refer Slide Time: 50:17)



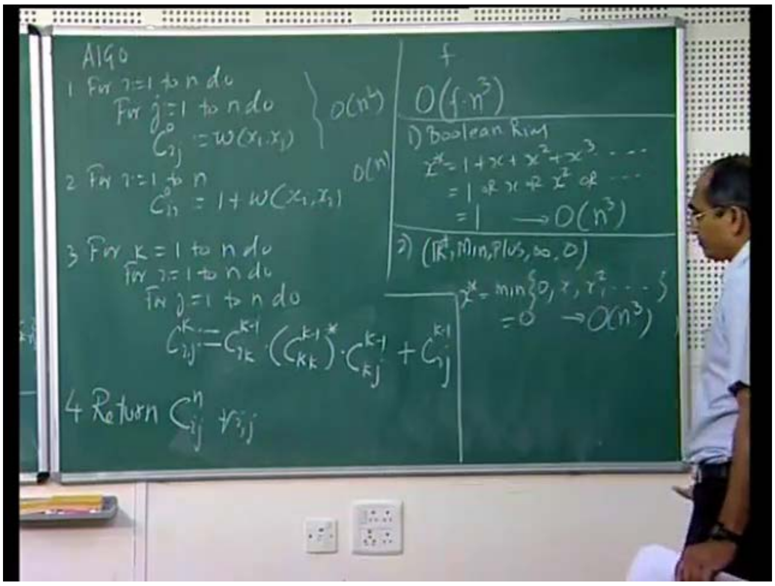
So, this will be C k minus 1 i j, plus C k minus 1 i k, dot C k minus 1 k j, plus C i k, k minus 1, dot C k minus 1 k k, dot C k minus 1 k j, dot dot dot. In general the term will look like, C i k k minus 1, dot C k k k minus 1 power p, dot C k j k minus 1. This is one occurrence of C k minus 1 k k, next time there will be C k minus 1 k k, dot C k minus 1 k k and so on. So, that can be written as the square of this the cube of this and so one in general ph power of this, and this infinite series proceeds let us simplify this expression now looks like. Let us write down as, C k minus 1 i j, plus C k minus 1 i k, dot 1 plus C k minus 1 k k, plus C k minus 1 k k, square infinite sum dot C k minus 1 k j.

(Refer Slide Time: 52:30)



This sum is written as, C i j k minus 1, dot C i, C k k k minus 1 star, dot C k j k minus 1, this stands for this expression this infinite sum. Now, I can simplify this expression and write this as C k minus 1 i k, dot C k minus 1 k k star, dot c k minus 1 k j. Now, we can replace this is the simple expression that computes the next level c expressions.

(Refer Slide Time: 53:09)



Student: C i j 0 should also.

C

Student: C i to c i to j k minus 1.

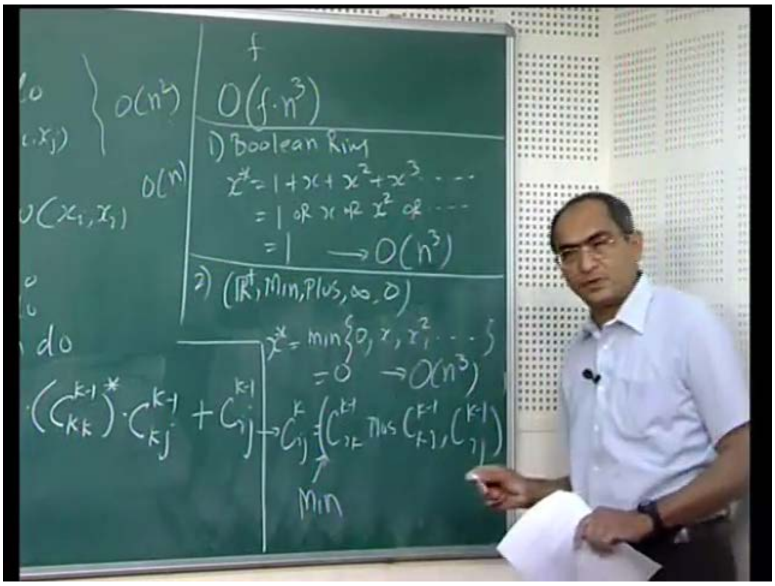
I am sorry.

Student: the path from i to j without using k.

Without using k is this, I have not written that; that is C k minus 1 i j that is right indeed. So, the fourth step is just to output C n i j for all i and j, the time complexity notice that this is order n square. This is order n the problem with the computation the complexity is that what is the time it takes to compute the star operator. If say it takes some time f then the time complexity will be order, f times n cube.

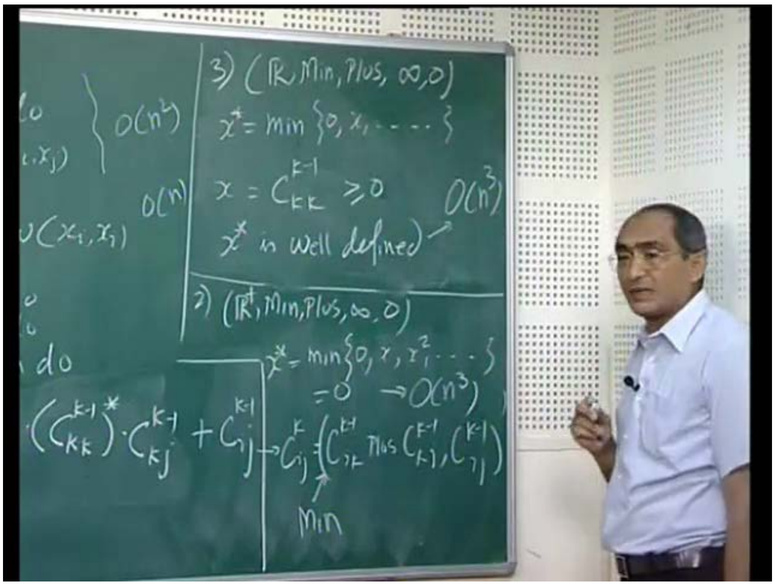
Let us quickly take a look at, what is f for our two examples, the Boolean Example in the Boolean ring x star which is 1, plus x, plus x square, plus x cube all the way is nothing, but 1 OR x OR x square OR and so on which is 1. So, this simply is replaced by 1 hence it reduces and simply it becomes C k minus 1 i k, dot C k minus 1 k j, plus C k minus 1 i j, so that means here the time complexity reduces to order n k. In case of the ring R plus min, plus, infinity and 0 the x star is the minimum of 0, x, x square which is 0. So, once again this is easy to compute and the time complexity reduces to order n cube, because you do not have to do any computation here so this 0.

(Refer Slide Time: 57:08)



The, expression in fact, let us write down will turn out to be C k i j, will be C k minus 1 i k, plus C k minus 1 k j. So, it is the minimum of the 2 and C k minus 1 i j it is the minimum of these 2 this is that simple. One more last comment is when we have this structure, which is really not or it is no longer R plus, it is R, min, plus, infinity to 0.

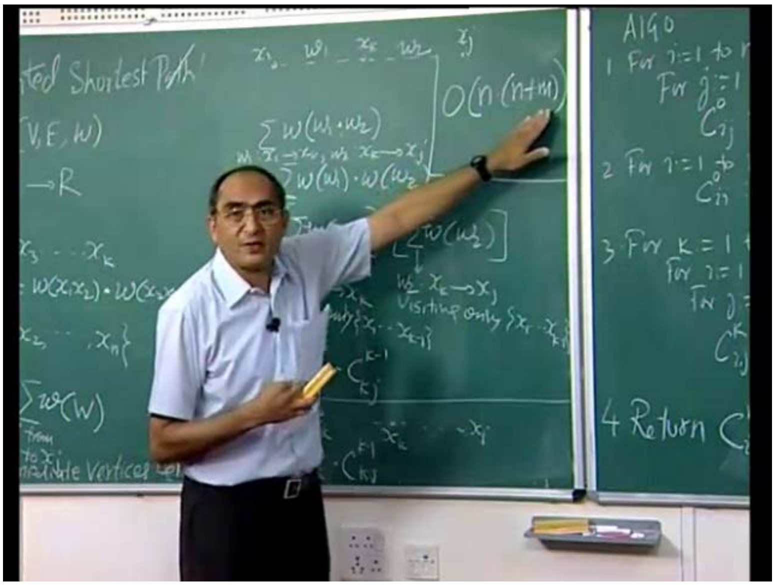
(Refer Slide Time: 57:55)



If you notice the only infinite sum that we deal with is this, so in this ring also it is well defined and x star is again it is nothing but the min of 0 x. Now, the problem will arise if this quantity is negative if it is non negative, then it is well defined. So, this weight also works as long as x is non negative. What is x? In our case x is C k minus 1 k k which is the total sum of the weights of the rings that start, walks that start and end in k so these are closed walks.

Hence this is because we have assumed that in our examples all the closed walks have non-negative weight so as long as this is the case, this is well defined x start is well defined and the time and it reduces as this case, and its time complexity is order n cube. So, the last point I want to make is that this approach unlike the Dijkstra’s approach allows us to deal with negative weights, as long as cycles do not have negative weight. But on the flip side the time complexity is n cube, but if you run the Dijkstra’s algorithm n times.

(Refer Slide Time: 1:00:12)



Then, the time complexity would have been n times, n plus m this will compute also all pair shortest paths. Note that m in the worst case is m square hence, in the worst case it becomes m cube there it is always m cube, so that is the positive side for the Dijkstra’s algorithm that is all.

Thank you.