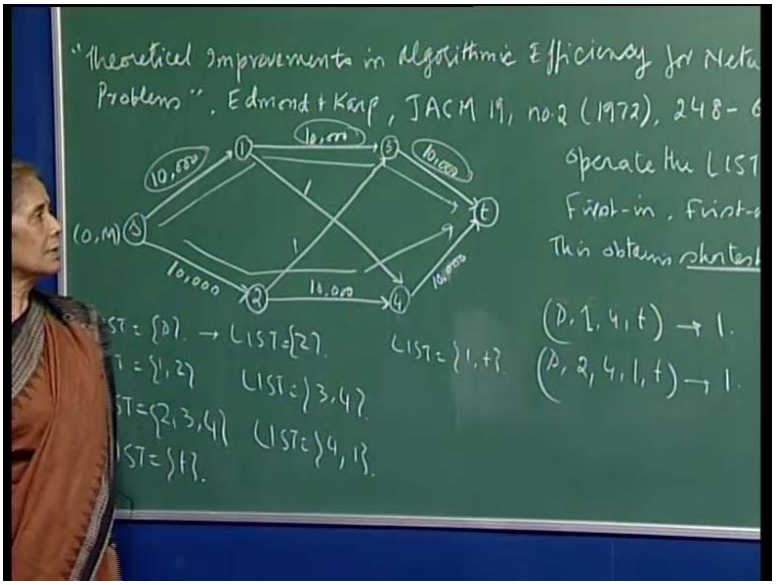
Prof. Prabha Sharma.
Department of Mathematics and Statistics.
Indian Institute of Technology, Kanpur

Module No. # 01
Lecture No. # 36
Improved Max-Flow Algorithm

In the last lecture I showed you the short comings of the labelling algorithm, the food in bulk essence labelling algorithm, so I thought that I will at least give you some idea about one of the improvements on that algorithm, because at this point of time food in bulk essence algorithm is not really used, because as I told you of the short coming. So, try because the scope of the course does not really permit a very detailed discussion of max-flow algorithms, I thought I will at least introduce you to one of the improvements that from mate some time ago. So, here the paper actually this was by Edmond and Karp, who have done lot of work in optimization, combinatorial optimization. And the paper appeared in 1972, in the journal of association of computing machines and the title was theoretical improvements in the algorithm, make efficiency for network flow problems.

So, this would be a very generic the kind of thing, I am going to describe to you is a generic algorithm, but when you applied to max-flow, you see the advantage. And again as I said, this is again only the first step of improvement, then later on they were many more improvements and it will be evident for you why this is not to the last word on max-flow algorithm. So, I will come back to the same problem that I discussed with you last time and so idea here is you see viewer the ford and fulkerson's algorithm does not say anything about how you operate that list least of nodes that you scan.

(Refer Slide Time: 02:00)



So, here the idea given by Edmond and Karp is that you operate the list as a…, so the idea here is operate the list as a queue and you mean you you know you understand what it means. That means, first-in first-in and first-out, so when you operate a list as a queue that means whatever came into the list the beginning, so according in that order only it will go out of the list. That means, you will pickup the, so here let me show you what I mean by that and this is also known as a brest for breadth first search and then because of breadth first search, we will see what the result is.

So, here, let see, you start with list as including node s, then when you scan s, your list will contain nodes 1 and 2, right, because you can from s, you can go to 1, you can go to 2, since there is no flow in the network right now. So, now, when you say first-in first-out, I will now take out number node 1 for scanning, ok. So, then in that case, so here, we I do not have to label, we understand what we are I am doing, so this is labelling 1 and 2. So, then when you operate, when you scan 1, node 1 from here, you will be able to label 3 and 4, right. So, your this will become, see this is what you mean by operating the list as a queue, right, so it will be 2, 3, 4, right.

Then you will now take out node 2 for scanning and from node 2 you can you see 4 is already labeled, 3 is already labeled, so you cannot do anything from here, so you will go to the next 1, right. From 2 you cannot label either 3 or 4, so I will go to 3 and from 3 you can label, so then when you the scan 3, you label t. And so a node t gets label and therefore your augmenting path is this, right. So you have seen that the capacity, of

course when you do the actually labeling you will also record the label from which, the node from which this you reached t.

So, therefore, you when you trace back the path, you also check the capacity and it comes out to be 10000, so you will say that the you have flown along the augmenting path S 1 3 T 10000 units of flow. So, therefore, this will become a flow along this path and now I start scanning again. So, you see here also S 1 as one augmenting path is found, they are going back to fresh labeling, so that defects skill stage that you are not um finding more than 1 algorithm augmenting path in 1 iteration. So, therefore, the later algorithms make an improvement on it, so here you see we will again start from here, then you can label, so here list will be this and then from here you cannot label 1, we can only label 2.

So, the second iteration, this will be simply 2, right and then from 2, so when you want to scan, node 2, let see, from 2 I go to 3 and 4, right, from 1 I can only label 3 and 4, right and then when you will scan 3. From 3 you cannot label anything, right, because the 2 is already labeled, so from 3 I cannot and I cannot label T, because this is already saturated, yes, right I could label from 3, I could label 1, right, because I can reduce the flow. So, when you when you scan 3, so you have 4 and then you have 1, because 4 is already labeled. So, the. So, from, no sorry not talking of 4, all I am doing is from 3, I can when I scan 3, I can label 1.

So, that is it, so it is come into the queue at the end of after 4, right and now when you scan 4, then your list will be t, sorry this will be t and 1, I should not forget 1, because 1 is already present. So, t or we can say again I should not spoil the order, because then the whole idea is lost, yeah. So, the proper would be proper thing the list would appear as 1 and t, yes and now, from 1, you cannot label anything, because all these nodes are labelled, so then I will go to t and so it and in fact, since t is there already in the list that means I have found an augmenting path and in this, this time, your augmenting path is this and this also has capacity 10000, right, because from t you came by a 4 and 4, you came by 2 and that 2 came by S.

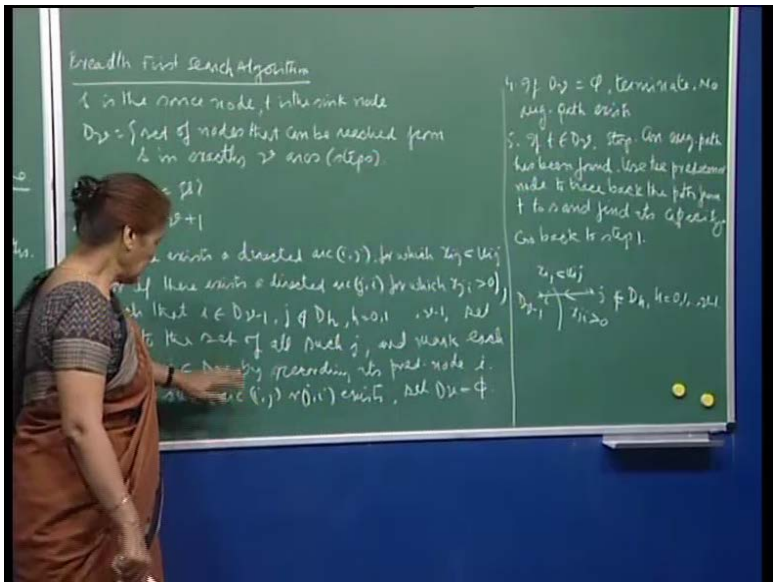
So, 10000 capacity and therefore, you have found the max-flow in two augmentations, just two augmentations and the total flow of 20000 units, so this is the idea. So, therefore, now I can spell it out that if you operate the list as a queue that means you use

first-in first-out search for your list for scanning the nodes, then you also look, you end up getting, so this help, this obtains shortest augmenting paths. And shortest means of course number of arcs, because we are not concerned about the distances of the arcs here, this only the number of arcs. So, you see this one, both these paths had three arcs in the shortest path and in the earlier thing when I had shown you. See for example, what were we doing when the labelling algorithm, see I had found for you, say for example, I found that path S 1 4 t, right, this was in S 1, this was S 4 4 T.

So, S 1 4 t and this had capacity of only one unit, right then in the next iteration we said we will we can look at the path S 2, because there is no harden fast rule about how you find in the ford and fulkerson labelling algorithm, S 2, then I can go to 4 1 and T, right. This was your a second augmenting path if you choose again with capacity 1, right and so the flow at the end of these two would be 2 and now if you alternatively, you alternate with these two augmenting paths, you will have 10000 20000 iterations, I sorry 10000, if you do take this as a fear, then you are having 2 units of flow in the network, so then you will have 20000 iterations to get the flow that you got by using the list as a queue in only two iterations.

So, the advantage is definitely there and now I want to formula is this algorithm, that is if you ah use breadth first search for looking for augmenting paths, then you can do a faster work and this this can be proved theoretically. And as I said again, the scope of the course does not really required that we talk about the actual complexity of this algorithm, but it will certainly be, we can prove that it is faster and say for example, the values of the arc capacities the u i j do not matter here at all, whether they are real numbers, integers or fractions, whereas in the ford and fulkerson algorithm we could show finiteness of the algorithm only when the capacities were either integers or rational numbers, ok. So, that is the point, so here later on when you are more interested you can go to further, (( )) more detail courses about these problems and then you can understand the complexity part and so on.

(Refer Slide Time: 10:49)



So, let me now come to this breadth first search algorithm. So, idea here is of course S is your source node, t is the sink node, then we will define d nu as the set of nodes that can be reached from s in exactly new arcs. So, we are now going to define a set d nu and we will say that the path from s to the node in D nu has exactly new arcs in it and it is very easy to construct such a set. So, let start with D 0 as the node s, then when use if you are working with nu then you go to nu plus 1, the next set and how do you constructed? If there exists a directed arc i j for which x i j is less than u i j or if there exists a directed arc j i for which x j is greater than 0 and such that i is in D nu minus 1.

So, that means, what we are saying is that suppose this is your set d nu minus 1, so node i is here and I am looking for arcs which are either like this j and node j should not be, see I am iteratively constructing the sets d nu, right. So, then if i is in d nu minus 1, then j should not b in any of the d h, where h varies from 0 1 to nu minus 1, that means it should not be in the set which have already been marked, ok.

So, I am so that means, here, if here j is such and if you have your x i j less than u i j, that means I can augment the flow on arc i j, then j does not belong to D h, where h varies from 0 1 to nu minus 1, this is the idea, right. And if the arc is, see i is here, but the arc is this way, right, from j to i, then I can only, again whole idea is I can label j if provided x j i is greater than 0, if this flow is positive. Then from going from i to j, I can reduce the

flow on arc i j and here again j should not be in the one of these sets d h, which is h varying from 0 to nu minus 1, this is the whole idea, right.

So, then I will put all such nodes j due to the set of all such j and mark each node j belonging to D nu, so I had constructed upto d nu minus 1, now I am showing you how to construct the set d nu and by recording its predecessor node. So, for example, for node j, I could label it from node i, so we note down the predecessor node, right and if no such arc i j or j i exist, then you said d nu to be empty. If I cannot find any arc i j like this, for which x i j is less than u i j or for which or an arc j i, for which x j i greater than 0, then my D nu would be empty and I will stop, I will say that no augmenting path, so if d nu is empty terminate no augmenting path exist, right.

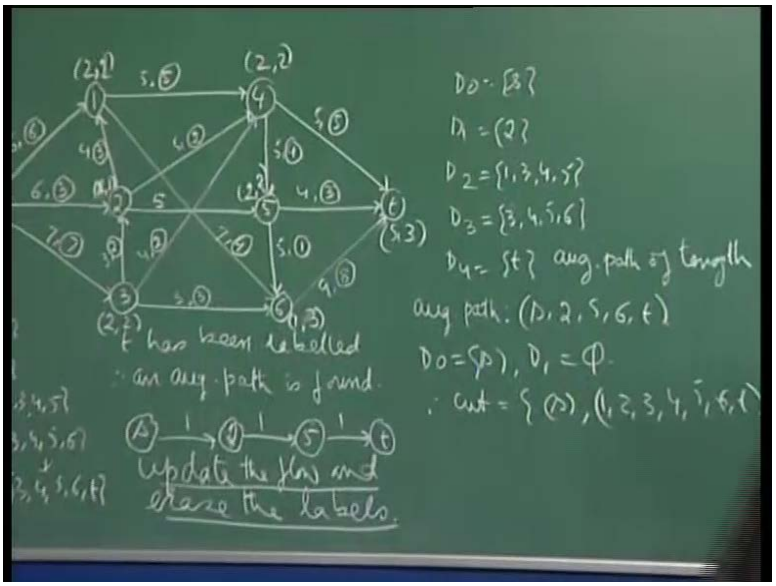
And in this case we will show that the current flow in the network is actually the max-flow, if t belongs to d nu then stop and augmenting path has been found, use the predecessor node to trace back the path from t to S and find its capacity, go back to step 1. So, then we will start all over again to find, see, so here, now, yes, a breadth first search again, I am I am giving this to you as an is a new algorithm just to you know, because the labelling algorithm was is not conservatively efficient. But, here, if you want to know why this will give us shortest paths then of course the you have to have a little idea of the the graph theory and the search techniques which again we are not talking about, but anyway it does not matter.

You can if when you work it out on a network you will see and when I show you the examples, one of the examples I will consider here, then you can see that yes this kind of search helps you to find the shortest augmenting paths. And then in one iteration, suppose you find shortest the augmenting path of lengths that say 4, then in the next iteration, you will find augmenting paths of you know 4 or 5 length will go on increasing and that is how it shows that the algorithm these are finite.

In fact, there are this algorithm will be efficient and more efficient algorithms can have also been constructed as I told you, because then in the in the newer algorithms you are searching for augmenting paths, more than one augmenting path in the same iteration. So, now, let me work out an example for you to show you the step, so I hope this is ok, so essentially we are saying that, we will always, whatever admissible arc, essentially in other words you can say that from node I all admissible arcs, then the corresponding

nodes will be set in the next set and then you will once you have constructed the set D nu, then we will go to D nu, a D the nu set. And because you will come back here, we will increase the index here and then we will again find all admissible arcs from the nodes incidence on the nodes in d nu. And finally, if either we find we are able to label t, how we are we say that we cannot the one of the sets comes out to be empty and then we stop, right, so let me work out an example here. So, let me show you the method that I had outlined breadth first search algorithm through this example and you see the flow is given to you.

(Refer Slide Time: 16:55)



So, the first number is the capacity u i j and this is x i j, the flow, currently flow in the network is this, let us starting labelling using the breadth first search algorithm. So, D naught, this will be 0, the label for this one for a starting node, then you see from here this is saturated, this is saturated, this is the only one, so I will label this as 1, so node 2 gets labeled, right. And so my list is be being prepared like this and we are saying that first come first-out, so this will be now from D from 2, I try to now label.

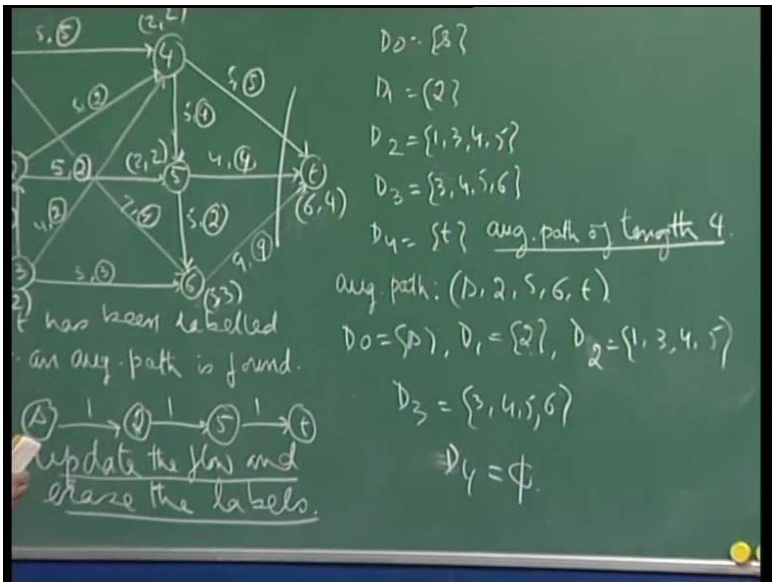
So, you see here there is a left over capacity here, there is a left over capacity here also and here, so that means you can label all the nodes and let me write it as 1, 3, 4, 5 and then we will now pick up the node 1 from here, so D 2 is this when I, so let me write the labels for all these S 2. So, 2 2 and 2 these are the label, right, now you try to start, you just pick up the node 1 from here, first-in first-out, so 1 you can you label anything from

here, yes you can label 6 from here. So, I will write 6 here, right, because 3, 4, 5 are already in the list, I will put 6 at the back, fine, that is all, then you start from 3, you pick up 3, you can you can label, 4 is already labeled, 6 is already labeled, so that is it. 2 has already been labeled, so we cannot do it, so 6 I will put the label as 1, ok.

We were also doing this that you put this and then you also put the level at which the this thing, the label has been put, so this was 1 1 actually, that is what we were doing, right, this will be 2 2 and this is also 2 2, because all are being labeled at level 2, there are in D 2. So, I should put this as 2 and this will also be 2 2, then when you label, now the third level labelling you start with 1, so this will be, this will be 16, 1 3 and here not done with level 3 labelling yet, because from 3 now you cannot label anything more. 4 also you cannot label anything more, from 5 you can label t, you see from 5 you can, because 4 and 6 are already labeled, so 5 you can label t and therefore the label for 5 would be, t would be 5 and 3, so third order label.

And the moment you label t, you have t has been labeled, so an augmenting path has been found, right and the augmenting path is S to 2, 2 to 5 to t and of course, we trace back because from t you will go to 5, right, from 5 you will go to 2 and from 2 you will go to, oh I should have put here, (()) from 2 you came to, for you you came to 2 from s. And the capacity then you will have to find out because you start going about, see here for example the capacity, left over capacity is one, then you will see, if you can flow that much unit back up to s, so, therefore this is the augmenting path with capacity 1, so we will now need to augment the flow, so let us quickly do it.

(Refer Slide Time: 20:37)



The flow becomes here 4, then this you have 1 unit of flow here and this becomes 4, right. So, then it says you erase the labels, this is again as I will show you the better algorithms exist which also do not say that you erase the labels the moment you have found 1 augmenting path, because that again is lot of duplication of work.

So, we will we will not be able to talk about that algorithm, but such algorithms exists, fine. So, now, we start labelling nodes again and you start with this as 0, then here again you can only label this, so this will be s and 1, level 1, right, then from 2 you can only label 1, 3, 4 and 5, because here again you have left over capacity here, here and here, ok.

So, I label these and of course the labels should be here, 2 2 you see, so therefore you are repeating so much work, so this would be because you have flow from 3 to 2, therefore you can label 3, because you can reduce the flow here. So, this will be 2 2, level 2 labelling, right and then 4 also gets labeled, 2 2 and this also gets labeled, so the same thing, you need not have reduced, erased all these labels here, fine.

Now, we will this is saturated from 5, I can label 6 and that is what we did here, so 6 came here from 5, 6 came to the list d 3. Actually I could have remove 3, 4, 5, there is no need, because d 3 is 6, right and then from 6 when you because here 6 again you can label t now, because there is some left over capacity. So, D 4 is t and therefore,

augmenting path of length 4, now, here the augmenting path was of length 3, it used 3 arcs, now it is augmenting path of length 4.

So, when you label again the path length will go up and the augmenting path is now s 2 5 6 and t, because here you came from, this was 5 and 3, right and the label for this became 6 and 4. So, 4 lengths, so the path length keeps increasing every time we label, so here then the cut, corresponding cut is s, because now when you want to label again, after you have, ok….

What was the capacity? Capacity of the flow is 1, so this will become 9, this becomes 2 and this becomes 2 and this is 5. So, please make sure that you have, so that means 11, 7, 18, 18 is the flow and you can see here also 9 plus 9 18 is the flow. And now when you want to label again, D 1 is not empty because D 1 is not empty because I will be able to label 2, that is still be there, fine, so I should not say this. D naught is this, D 1 is 2, again numbers are, I do not know may be the initial flow 18 and so then can you again find, so now you will not be able to find, yes.

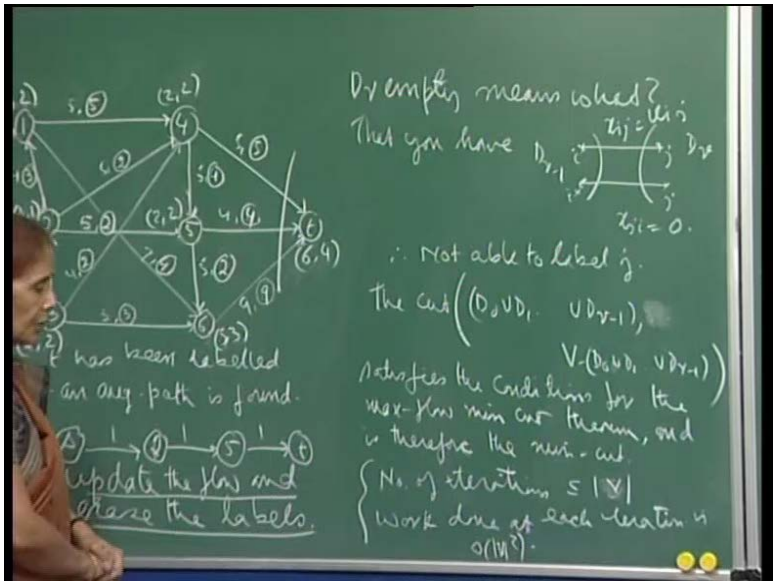
Now, we try to label, we will stop here, but D 1 is certainly not empty, now from D 1 can you label anything, from 2 can you label, yes, you can label 1, you can label 1 and then you can label 4, again you can label 3 and 5. So, how did I write, so therefore this is not correct, that means your D 3, D 2 is again 1, 3, 4 and 5, because all these numbers they are from 2, you are able to label 1, you are able to label 4, 5 and 3, because here again this is it, right, fine. So, then fine.

So, now we want to label from 1, I cannot do, I can label 6, so here again D 3 will be, so I will write 6 at the end, 3, 4 and 5, I should keep removing the nodes which I am not using, right and then again you want to label something, can you label something from 3? Can you label something from 3? 4 is already label, 6 cannot be labeled, everything is done, so 3 is gone.

Then from 4 can you label anything? You can label 5, is already labeled, 4 you cannot label, anything then, 5, 5 you cannot label this, you can, 6 is already labelled and that is it, so this is it. So, now this, therefore your D 4 is empty, D 4 is empty and so the cut, remember we would say that the cut is all union of D naught, D 1, D 2, D 3, because the list you keep dropping the nodes we are from which you have already, which you have

already scanned. So, D 4 is empty right and therefore your cut is actually, your cut is actually 6, so it will be s 1, 2, 3, 4, 5, 6, so this is the cut this is the cut because your cut, right remember we said that you add up all these node, these sets and then so this will give you and because now you do not have anything, any valid arc from 3 to 4, because they are all saturated and this is what we will just write down and show you the validity of this, but this is a verification and so, therefore, the current flow in the network is 18, the max-flow in the network is 18.

(Refer Slide Time: 27:11)



So, just want to give the last comments on this and I have already shown you through this example, what we are saying is that the moment see we said that here that when you hit there is a set to be empty, the D nu set is empty, then you stop and see what is happening, why is D nu empty, you can just see that from node i in D nu minus 1 whatever arc is there, i j is a forward arc, then you see its already saturated. So, therefore, you are not able to know, label node j right and so you cannot find any node in D nu this way, that means which is connected to node I through a forward arc.

Similarly, there is no j which is connected to node i in D nu minus 1 via backward arc, because the flow x j i is 0 and therefore i cannot label j from i, since what is happening and so my D nu is empty, right, I am not able to label any more nodes from nodes in d nu minus 1 and so D nu is empty, right. And you can see that the cut would be, you add up all the D naught to D nu minus 1, add union of all these sets, then this and V minus this

set, this will give you the cut, see thing is that D 1 is only connected to arcs in D 2 and so on. So, essentially no problem because you do not have any arcs from here, two nodes here, right, you only have arcs from D nu minus 2 to D nu minus 1 and so on.

So, this is a cut and the cut-set will of course will be the edges like this, which are either forward arcs from nu minus 1, nodes in nu minus 1 or arcs coming into to nodes in nu nu minus 1, but the arcs coming into have flow 0 and the arcs going out of nu minus 1 are saturated and therefore this satisfies the conditions for the max-flow min-cut theorem and so this must be the minimum cut, right. And now you see that as I showed you through the example of course, these two things, we will not this is not part of the course to be able to show you that, but you can see through this example that because of the labeling, when you exhausted the possibilities of labeling, finding augmenting paths of a certain length, the next time when you started labeling, the length went up, right from 3 to 4 in this case.

So, here also, therefore, you see the number and since the path length cannot be more than n minus 1, therefore the number of iterations will be bounded above by the number of nodes in the network, right and then you can also show it is possible to show that the amount of work that you do at each iteration, because you are after all just trying to scan all the edges which are either coming in here or going out, either going out or coming into this. So, then the amount of work that you would do will not exceed order V square the number of node square and so the total complexity or the the time require to work out, that means a number of a basic steps that you would need to work out this algorithm would be into order V q.

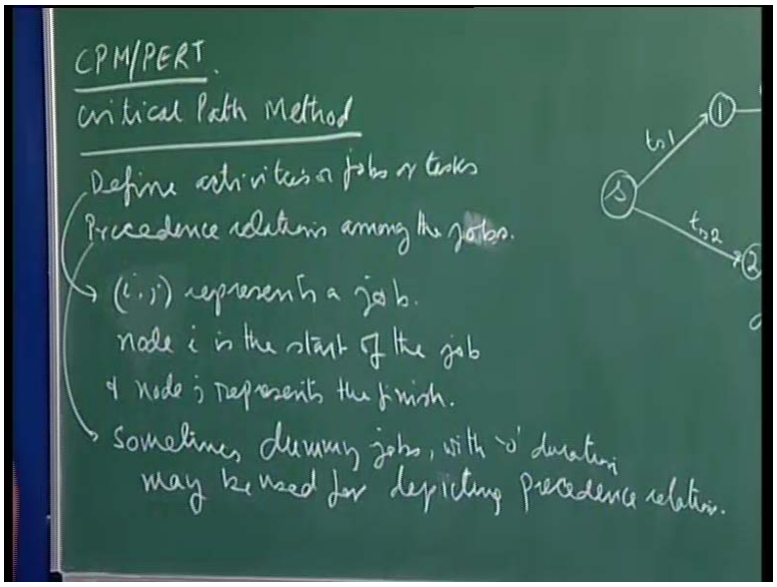
I just giving you an idea of course, we are not going into all those things, so this is the whole concept behind, trying to improve on the ford and fulkerson labelling algorithm and then as I said they are more refinements on this, which make the algorithm more simpler to implement. So, let me now talk to you about topic pert and c p m is the this is called critical path method and program evaluation review technique. These are used this is you know these problems arise in project managements, you have huge projects, say for example, at some point it was a sending a man to the moon, it was a huge project, then you can imagine that such a project would entail thousand different jobs which could be done you know in small teams independently and then you know, but whole thing had to, the whole effort had to be coordinated.

And you can imagine a project manager or a who has to oversee the whole project, anything of say that magnitude a project would no cannot just work without having some support in the form of so therefore, in in the form of the techniques that we are going to talk about today .

And my reason for including this in this course is that, you can see that some of the network techniques that we have learned and the linear programming formulations, we can can help you to you know formulate these problems in a way so that you can try to find solutions, then you can provide some short of guidance to a project manager who would otherwise be simply helpless, because he will not be able to, you were she will not be able to otherwise take care of how something something is going wrong, something is getting delayed or something can be you know done in a faster way, then the whole ideas the project should not get delayed unnecessarily.

And if you have schedule somehow if you can manage to keep up to the schedule and all these aspects are there. So, I will try to just give you a few examples so that you understand where these situations can occur, so here the first I will talk about a Critical Path Method.

(Refer Slide Time: 32:43)



So, this is the critical path method, so a large project get divided into activities which we think are sort of in a way independent, that means different sets of people or different

resources can be given to these small task or jobs which can be done simultaneously. But, obviously you need, so what are the things that you define, you have to define, your define activities or jobs or tasks, whatever name you want to give to them, right. So, you have to first define these and then you have to have some sort of precedence relations among the jobs.

And after very good precedence relations among the job, say for example, if you are building up or let say a tyre has punctured on the way, you are going somewhere and a tyre gets punctured, so you cannot for example replace the tyre or the unless you take out the stepney, from the from the dicky from of the car, right. And you cannot put the stepney unless you remove the puncture tyre, the wheel from the first you have to remove the wheel from the car and then you have to remove the, then and replace the stepney, put the stick, the stepney to the car and so on. So, the job, the sequence of jobs is also fixed in a way.

And here I am talking of a situation where sort of the jobs are definitive, we know exactly what resources, how much time and so on the jobs will require and so here critical path method is mostly used for situations where a situation to the things are very deterministic. So, precedence relations among the jobs are required and you I would like to use the presentation for these critical path method problems by saying that you know for example a job or an activity will get denoted by appear of nodes, right.

So, for example, so here first of all when you define i j, I represent to the job i j, so this i j represents the job, node i is the start of the job, right and and node j represents well maybe it is enough to say that node i, node j represents the finish. It represents the finish, therefore, for example yes suppose I have an arc like this also here, so let us look at this this network here representing a project, I have not shown the finish here, the finish may be somewhere and then you may have something like this, this may be the finish. See here you start and what it is saying is that activities s 1 and s 2 can begin at the same time, because the resources and the kind of team that you require for starting these two jobs their different they are independent and so you can begin the project from this point of time and both these activities can start at the same time, right.

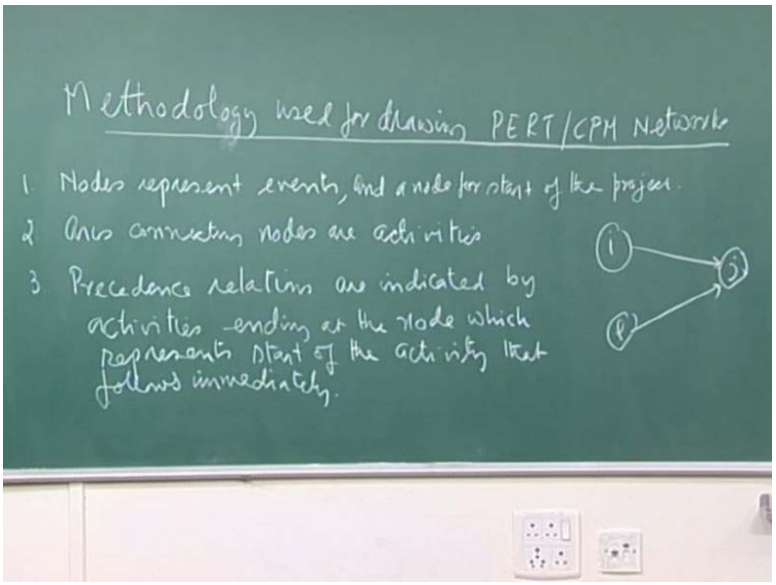
Then t s 1 gives you the, gives you the time to complete this job, t s 2 denotes the time to complete the the job s 2, right and then now I am trying to show you the precedence

relationship. So, what it is saying is that job 1 3 cannot begin till job s 1 is complete, so this is the kind of precedence relationship. And here, for example, in this case, we are saying that node 3 is the beginning of job t 3 4, of job 3 4 and of 3 5. So, when I say that for node i j, if you have this arc i j, the node i is the start of the job and then it can be the start of more than one job, right.

So, normally a nodes are treated as events, that means at node 3 this is the beginning of both the jobs 3 4 and 3 5, and when I do this broken arrow here, that means this is a dummy job, this is a dummy job, because in this network I want to reshow the, depict the relation precedence relationship that jobs 3 4 and 3 5 both require. That job s 2 should s 2 and 1 3 should get completed, both the jobs have to be completed before jobs 3 4 and 3 5 can begin. So, therefore, by doing this and saying that it requires no effort, no resources, so I am putting a 0 here, that means the duration of this activities also 0. So, therefore, this now shows that 3, the jobs the even 3 can take place only ones the jobs 1 3 and s 2 are completed and then this two. And so what I was saying is that here for example, at this point, I can say that this is also the finishing point of time of job 3 5 and 6 5, but actually node 5 denotes the finishing of both the jobs 3 5 and 6 5. So, this kind of this clarity has to be there, right and then of course, so they you have activity 3 4 and then this you can call it, say this will be 6; this will be 7, node 7.

So, then, all these activities when they get completed then you have the project gets completed, right. So, here, this is the representation and the precedence relationships, will be depicted by this kind of a node arrow arc, node and arrow arc. And then precedence relationships among the jobs has specified, then here sometimes dummy jobs with which with with a 0 duration may be used for depicting precedence relationship, precedence relationship .

(Refer Slide Time: 40:24)



See while discussing pert c p m, I did give you an idea how we would go about drawing the network, but here let me give you a formalize the whole thing. So, we are using the representation where the nodes represent events and so we will have a node first, will have one single node to depict the or represent the start of the project. And then, of course, the active the arcs connecting nodes are activities, for example here i j will represent activity i j which begins at node i and set node j, similarly activity l j will also will begin at l and end at node j, right.

So, arcs connecting nodes are activities and then precedence relations are indicated by activities acting at the node which represents start of the activity that follows immediately. So, where example if you have the activity some j k then j k activity cannot begin unless both activities i j and l j end at node j, so, therefore when two arcs are more than whatever the number of arcs, which end at a particular node, they all represent the activities which are immediate predecessors for the activity which begins at node j. So, this is the kind of precedence relationships we depict in the network and then we often use dummy activities to show precedence relation.

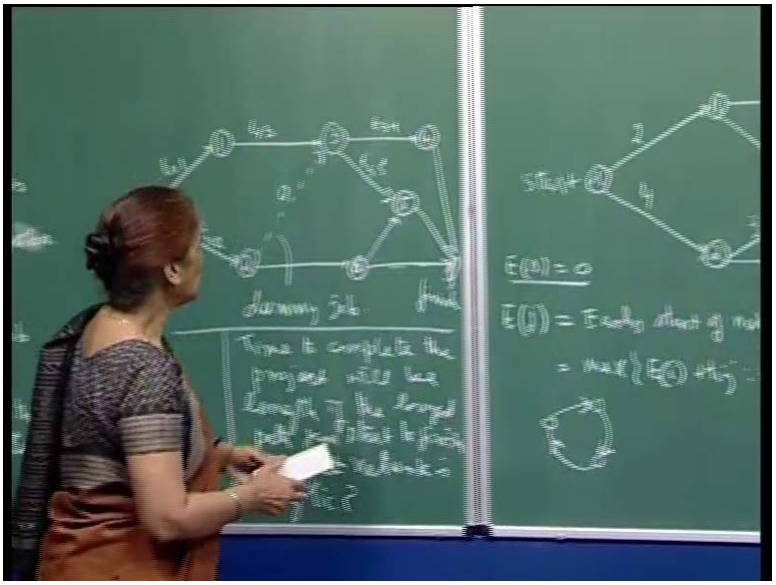
Now, for example, for activity j m, activity i j and i k both are predecessors, but then since activity i k is also a predecessors for activity k n, so, therefore, in the network I have joined k n j by broken arc, which denote which denotes a dummy activity. So, now, the network shows that for activity j m activity i j and i k both are predecessors,

immediate predecessors. So, this is how and of course in the beginning when you are drawing the network, you can use the dummy activities freely and then again you can revise the whole thing and see if you can drop, down drop even a cut, emit, omit some of the dummy activities, right.

Then numbering of the nodes is done from top to bottom, so essentially what you do is you first draw the whole network showing the precedence relationships and so on and then you start numbering the nodes from top to bottom, and then their should be a node indicating the end of the project, fine. And then of course, as we discuss the problem later on I also show you some of the ways how which we manage the network, ok.

So, this is the this is therefore we can have this node arc representation and this sort of gives you feeling of the flow that how the work is progressing and then you can chat it out and you can see it from day to day how the arc from month to month, whatever your time unit, how the project is progressing. So, this gives an idea to the project manager.

(Refer Slide Time: 43:33)



Now, the refinish time you can see here, yes, so this will tell you the when when when you reach t, that means the the time, a time to complete the project will be the length of the longest path, longest path from start to finish due to… the remember I told you that we would need to compute longest path in a network. So, now, let me make two comments here, so first of all a project will be completed only when all the jobs have

been completed, so which makes, so therefore it immediately realize that the longest path from s to t will give you the duration of the project, right.

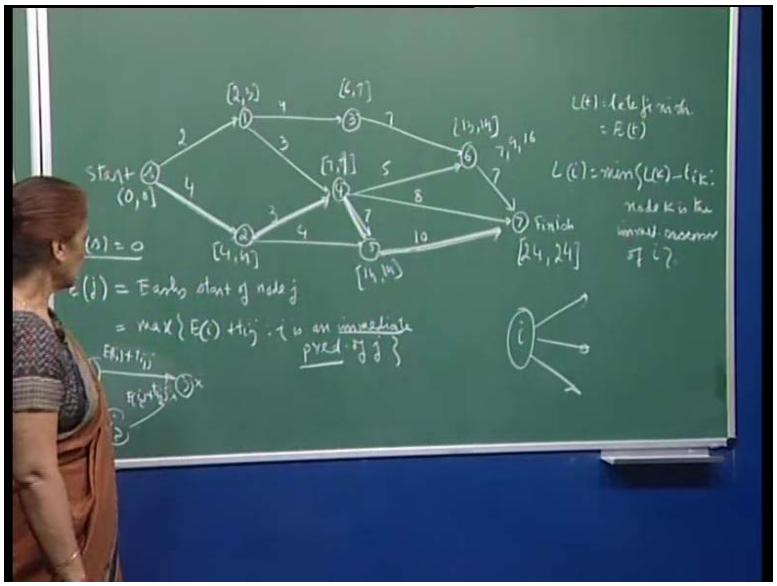
And secondly the this network is a cyclic, the net the project network may be, I can the project network project network is a cyclic, why? Why why is it a cyclic? See, if if it had cycles that means what? That means your precedence relationships are quality no, if you have, if you had a cycle like this in your project network then that means what will it say? It will say that this job must be complete to be, for this job can be completed, but then this is the immediate predecessor of this, which means that this job can only be, that this this job can start only when this job has been completed, right, but this job could not start before this what is completed. Now, here, if you have this kind of cycle obviously the precedence relationships that you have defined are not valid.

So, the presence of when you start drawing the project network, the moment you encounter a cycle you will realize that your precedence relationships are wrong, there is something and so you should go back and revise them and then draw the network again. So, these are, so, therefore and once you know that the project network is a cyclic there is no question of a positive cycle present in the network and therefore, longest path can be found and you know that you can just do the minus of the a c i j and here the c i j and you will be able to apply the shortest path techniques, fine, right, ok.

So, now, just to show you if you computations how to be find out the longest path, so here because the a project is a cyclic, the finding the longest path becomes very straight forward and so I will show you that just by simple labeling. We do not really how to go back and change the t i j to minus t i j and apply that express algorithm anyone of them, because we can do a fastest job here. So, I would like to remark here that see all along this course you have seen that we have the theory develop for solving a certain class of problems, but when you have special features present in a problem, it helps to exploit those special features and come up with faster or better algorithm. Since, so that is what is happening here to find the longest path, we will show you the quick way of doing it. So, here let me start, let me the start defining e j, is the early start of node j, so now I will not refer to the job here, right now, because as I told you for example, node 4 here it is the start of job 4 6, the jobs 4 6, 4 7 and 4 5.

So, I will just simply say early start of node j, right, let me define e j. So, which means that here we will always say that e s is 0, let we start, the clock begins from 0 time, right and then we will use this definition that for any node j you want to find out the early start. And obviously, for example here, this node cannot begin, the start for this node cannot begin unless job jobs 2 4 and 1 4 have completed, right. So, therefore, the definition would be max of E i plus t i j, where i. So, yeah i should have may be define the word immediate predecessor. So, this is immediate predecessor means that, here for example, for 3 immediate predecessor said is 1, right, or you can say that for job 1 3 the immediate predecessor is s 1 in terms of jobs, so you can have both concept of immediate predecessor node or predecessor job, right. So, here and so for example, for 5, both this and this, 3 and 6, both are immediate predecessors for node 5, right. So, when you want to find the early start for node j, then what you want to saying is here so you have a node j here, ok.

(Refer Slide Time: 49:00)



And you will look at all the immediate predecessors here, i 1, i 2, see for example and then early start of this plus, so this would be E i 1 plus t i 2 j, similarly this will be E i 2 plus t i 2 j, so this will be t i 2 j. So, you will take the max of the two, because remember this node cannot begin unless both the activities which are coming into the node are completed. So, therefore, this will be max of E i plus t i j and i is an immediate predecessor of j, so let us quickly write down, see for example here there is no problem, because this is 0, so start is 0, so I will fight the numbers as this and then I will complete.

So, here this will be 2, yes, because this is 0, so this is 0 here and then this, so similarly here it will be 4, yes, right. And now let us come to node 3, so node 3 here again, there is only this predecessor, so this is no question of max, so this will be simply 6, yes and I am not completing the other part which we will do after I define the…., fine. So, now, for node 4, for example, this way it is 5, right, this 1 and here it is 7, so I have to take the max. So, this will be 7 and then for this one, for example, this is 4 plus 4, 8 and here this is 7 plus 7, 14.

So, again, I take the max, so this get this node cannot begin before if you will toughing in terms of days. Let say for example, on the 14th day only, because you have to 14 this early start of this, right, then if you now come to node 6, then here this is 7 plus 6 is 13 and 7 plus 5 is 12. So, 13 is the larger number, therefore this is this and finally you complete the finish, so the finish part here will be, we have to add 10 plus 14, 24, then here this is 15 and I did not write down the 5, 6, 7. What is the number for 6 7 does not matter, maybe we can write something here and then we can correct it, yeah, maybe it 7, right, then this will be 20 and this is 15 and this is 24, so this is 20, ok.

So, this gives you the early start of all these nodes and so the earliest job project can finish this on the 24th day and similarly now thing is that I can also now talk of the late finish and then the late start for each node. So, then the calculation would be backward, and I will say that, let say that this is equal to early start, I am using the node t, usually again the same concept, sink and source, also source and sink is used, I have written 7 here, but.

So, if you take this fact that a late finish is also the same as early finish, given that or sometimes what happens is there is a due date for the project, so then the late finish can be taken as the due date of the project and then you compute backward and the computation is again simple and straight forward. Here, now, you will be taking the minimum, but then it will be immediate successors, see what we are saying is that if you have a node I here and you want to compute it is late finish, then you have jobs like this right and so you because all these jobs have their late finish, this nodes and then you would subtract the completion time of the (( )) of these jobs and take the minimum. Because again you see, this node, the latest finish can be only when all these jobs have completed, so the completion times of this and the late finish of these nodes and then minus the processing times of these jobs and then you take the minimum, right.

So, for example, here I will just quickly take, do computation for 1 node, see 24 and if this is 7, so here you see this is 24 minus 7, so this will be 14, you get my point. Because, here, this node just only one successor and this is 24 is the late finish, so minus the completion time of this job, so that is 14. But for example, for this one when you do it, so here, for come come to this point, so this is again 24 minus ten, which is 14, ok.

And for this one it will be, yes, so here now you have to do it this way, at this point it is 14 minus 5 which is 9, so one number is 9, then this is 24 minus 8, this is 16, can you see it 24 minus 8 is 16 and then, so that is it. So, these are the two immediate successors of job four and so you have to take the minimum, which will be 9 here, sorry, sorry and then sorry, for 4 they are 3 immediate success 6, 7 and 5. So, 5, it is 7 minus 14 minus 7 which is 7, so actually the minimum of the… So, therefore, this will be 7 and similarly when you go back here, you will see that this will come out to be 0, so here this is 7 minus 3, which are 4.

And here, because only one immediate, two immediate successors here, yeah here, you should have computed this 14 minus 7 is 7, right and then here this is only two immediate successors, so this is 7 minus 4 is 3 and 7 minus 3 is 4. So, the minimum is 3 right and then this is 4 minus 4 is 0, so, therefore, of because here this was 1, so minimum again is 0. So, this is your computation for the late finish and early finish and in fact, at least you see the what was your path, that you found out. I will quickly just see 24, so this was this, then you had this and we had and this, this was the longest path, quickly 7 7 14 plus 10 24, this was your longest path and so we call this the critical path, because if any activity on this path gets delete, then your project gets delete. All others you see there is some sort of slat, which I will define in the next lecture, but this is the concept. So, the longest path would be found by simple labelling algorithm, which till take long time and then because the algorithm because the graph, because the network is a cyclic and so now we will a show you how to do some more analysis for this problem.