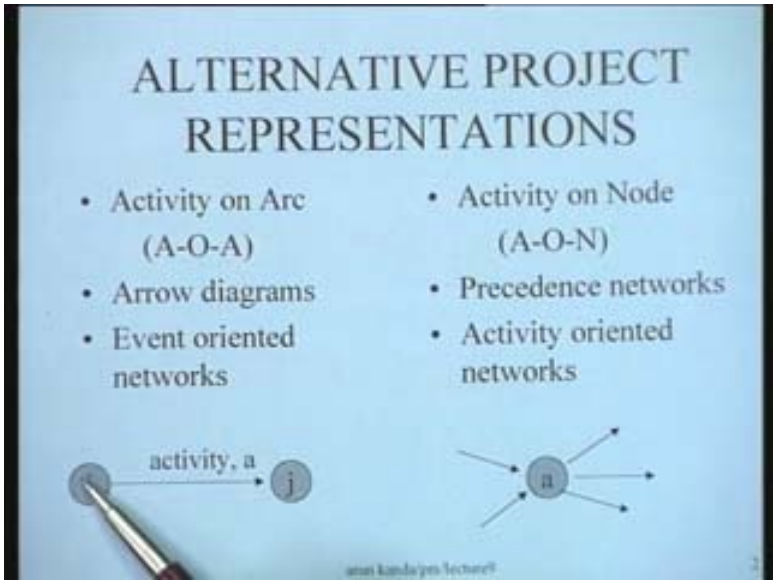
**Project and Production Management**
**Prof. Arun Kanda**
**Department of Mechanical Engineering**
**Indian Institute of Technology, Delhi**

**Lecture - 10**
**Basic Scheduling with A-O-N Networks**

Today we are going to be talking about basic scheduling with A-O-N networks. In our last lecture we had talked about basic scheduling with activity on arc networks. Today the focus is activity on node networks. You already know the difference between activity on arc and activity on node networks. They are alternative project representations. In the activity on arc representation which is also known as arrow diagrams in common parlance or also known as event oriented networks what happens is that the job is represented on the arrow and the nodes have the interpretation of events.

(Refer Slide Time: 1:47)



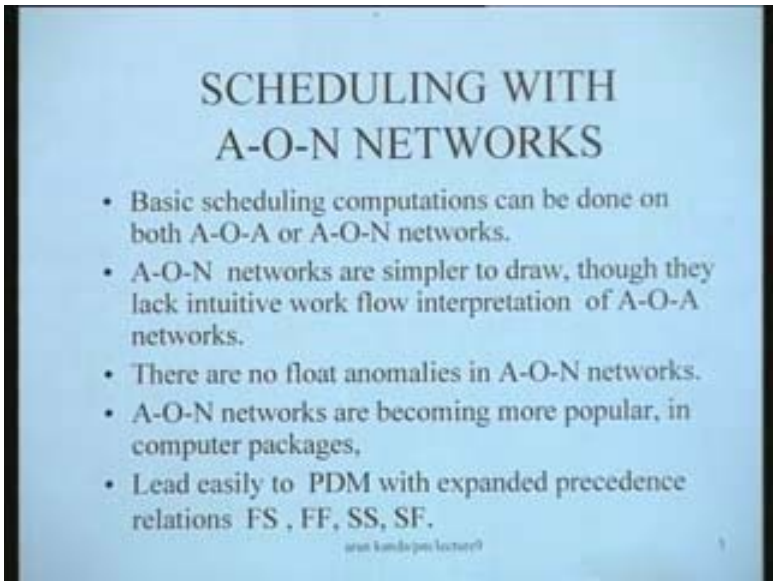

That is why these are called event oriented networks and the forward pass and the backward pass computations in A-O-A networks actually have the interpretation of event times or milestones here. On the other hand in the activity on node representations which are also known as precedence networks these networks are actually activity oriented networks. Each activity or the node here represents the job and the arrows are representing in fact the precedence relationships.

There are certain features of scheduling with A-O-N networks. For instance one can see that basic scheduling computations, that is determination of the schedule and the critical path for the project can be done on both the A-O-A and the A-O-N networks. The A-O-N networks are generally much simpler to draw. The A-O-N networks do not require any logical dummies and they are much simpler given the activities and their predecessor

relationships you can immediately draw the network without any problems and that is one reason why most computer networks or computer packages like MS projects and others tend to draw A-O-N networks or precedence networks. However the disadvantage is that they lack intuitive work flow interpretation of the A-O-A networks which means that in the A-O-A network as you progress along the arc it is like doing the job; it's like job being started to the job being completed. It's like an intuitive work flow interpretation as you progress on the network. That kind of an interpretation is missing in the A-O-N framework.

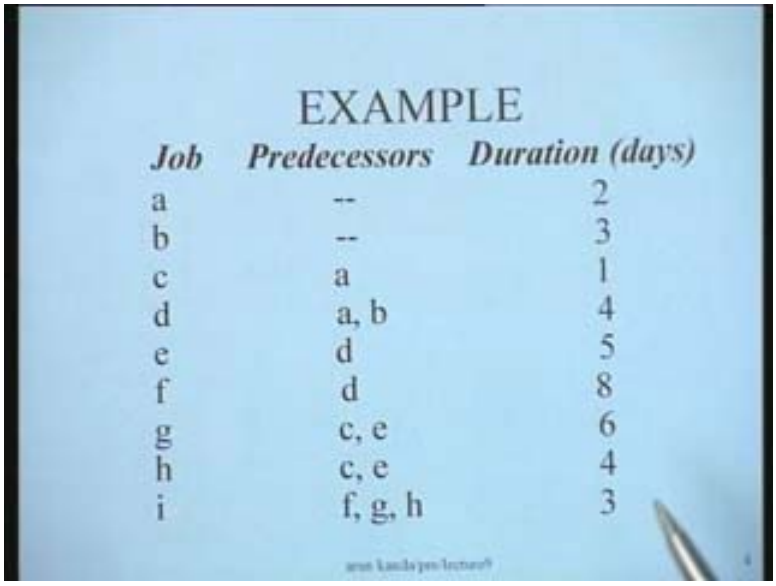
However we also noticed that there were some kinds of anomalies in using the 4 floats when we represented the project as an A-O-A network. This depended upon where you place the dummy whether the dummy was preceding the arc or the dummy was placed after the arc. However there are no such float anomalies in A-O-N networks. A-O-N networks as I said are becoming more popular in computer packages and these networks lead easily to a method of programming known as PDM or precedence diagramming methods which actually work with expanded precedence relationships rather than the conventional precedence relationships and there are 4 types of relationships finish to start, finish to finish, start to start and start to finish. With these additional lead lag factors we can do scheduling with A-O-N networks.

(Refer Slide Time: 4:52)



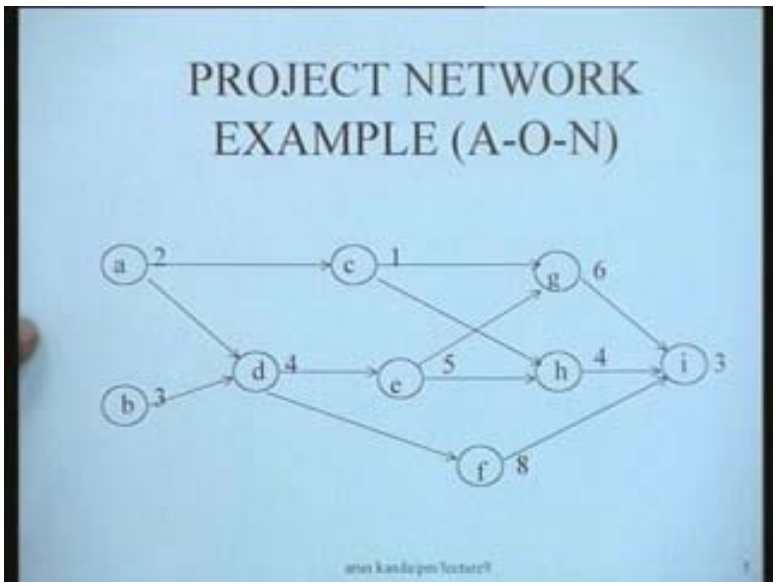

Let us now take an example and see how scheduling can be done with A-O-N networks. Here is a project which has this job description. This is the predecessor set and the durations in days for each of the jobs a, b, c, d, e, f, g, h and i are given as shown here.

(Refer Slide Time: 5:21)




From the predecessor information we can draw the A-O-N network without any difficulty because there would be a node corresponding to each job and the predecessor relationships would in fact be shown on the network by means of arrows. In this particular example the A-O-N network can be drawn as shown in this particular diagram.

(Refer Slide Time: 5:55)



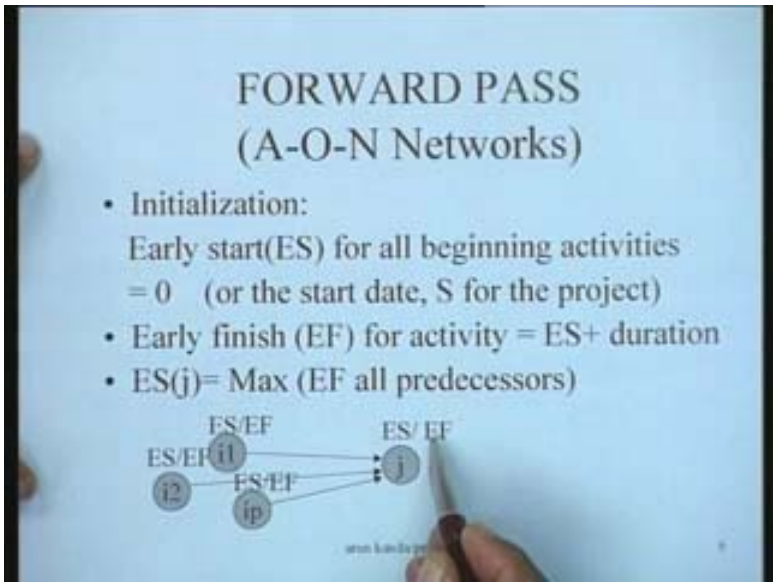

What you see here is that a and b are two jobs which have no predecessors and as far as job d is concerned it has predecessors a and b and these are shown here and similarly the predecessor of job c is job a. So you have an arrow from job a to c in this particular manner and then we have the complete A-O-N network as shown in this particular

diagram. You will notice that we have not used any dummies at all in this representation and what we have in this particular network is we have 2 source nodes. These are 2 source nodes because they have no predecessor arcs and similarly this is a sink node because we have 3 arcs coming into the node i but there are no arcs emanating from node i. There are 2 sources, 1 sink and the numbers adjacent to each node here represent the arc duration.

Job a has duration of 2 days. Job b has duration of 3 days and so on. The job durations of each of the jobs are in fact shown in this particular network. What we are now going to do is basically perform the forward pass and backward pass computations on this network. Notice that we do not have an event interpretation at all. We have the jobs which have durations which are given here. What we are actually going to compute is the early start and the early finish of activities which is obtained by a forward pass and the late start and the late finish of activities which is obtained by a backward pass. Before we actually do the forward and the backward pass for this particular network let us see what would be the general manner in which a forward pass would be conducted?

For any network a forward pass on an A-O-N network begins with a stage of initialization. That's the first step and initialization is actually the step where we say that the early start or ES for all the beginning activities; beginning activities meaning those activities which have no predecessors or the source nodes. This is equal to zero or it is equal to the start date S of the project as the case may be. This is the process of initialization for the forward pass. Having done this what we then do is we make this computation. Knowing the early start of any activity the early finish of the activity can be computed simply as the early start plus the duration of the activity. It's as simple as that. The early finish is equal to early start plus the duration of the activity. How we would be computing in general for an activity j, the early start and the early finish?

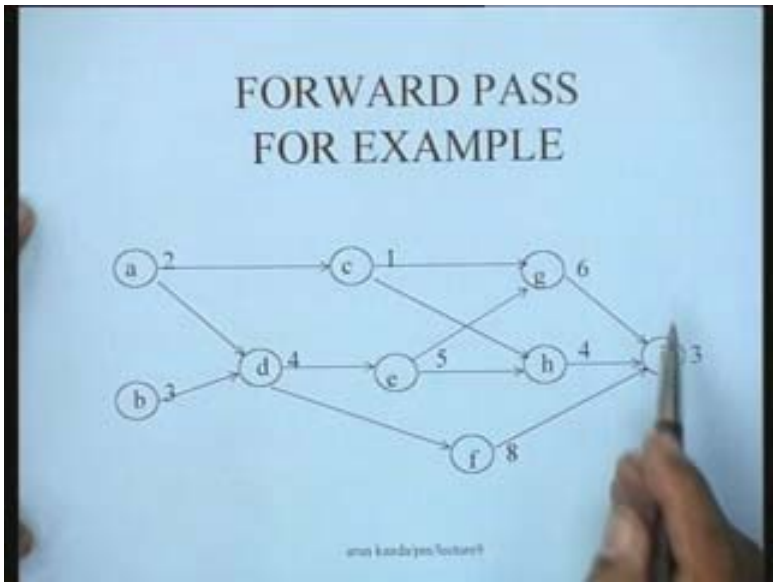
(Refer Slide Time: 9:18)



The early start and the early finish are computed for activity j simply by looking at the predecessor activities. Suppose this activity j has 1, 2 and so on up to p predecessors. Then we have already computed by virtue of the previous steps the early start and the early finish for each of these activities. What we have to simply do is we have to take the early start of activity j as the maximum of the early finish of all predecessors. You look at this number which is the early finish of these particular activities and we compute the early start for this activity. Once the early start is computed the early finish is then computed by adding the duration to this activity. We can obtain the early start and the early finish for a forward pass.

Just to illustrate this procedure let us go back to our example and see how we would be doing a forward pass on the network. Going back to the example let us now do a forward pass for the example. This is our network.

(Refer Slide Time: 10:25)



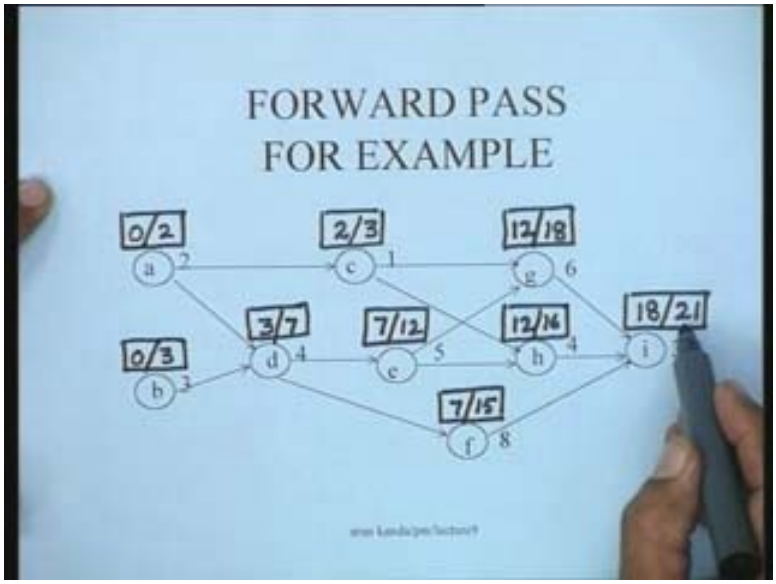

Let us take for instance the initialization step for this particular example. a and b are the two activities which do not have any predecessors. So they are the beginning activities. They can start in fact at time zero. This can start at time zero and this also can start at time zero and moreover this particular activity has a duration of 2. So its early finish will be 2 and the early finish for this activity will be 3. I can put these in a rectangular box at the top of each of these activities. Remember that the first number is the early start and the second number is the early finish of the activity.

Let's take activity c. Activity c has only 1 predecessor. So there is no problem. The early start of this is going to be simply 2 and since the duration is 1 its early finish is going to be simply 3. The early start and the early finish for activity c in the forward pass is simply going to be 2 and 3 respectively. Let's look at activity d. Activity d has 2 predecessors. What you have to do is simply look at these early finish times. The larger of the two is 3. This is simply 3 and the duration of this activity is 4. So 3+4 is 7. The

early start and early finish of activity d can be determined in this manner. It is 3 and 7. Let's find out the early start and early finish for these subsequent activities. For activity e it will be 7 and 12. This is the early start and early finish for activity e. For activity f it is simply going to be 7 and 15. This is the early start and early finish of this. Take activity g; activity g has 2 predecessors. So you have got to take the maximum of 3 and 12 in this particular case. So this is going to be 12 and it has duration of 6. The early finish for this particular job is going to be 18. 12 and 18 is the early start and early finish for job g.

Job h has these two predecessors. The larger of the two is 12. 12 and 16 is going to be the early start and early finish for job h. We come to this terminal job i. It has 3 predecessors g, h and f and the larger of the early finish values which are already computed is 18. So this will have an early finish of 18 and then it has duration of 3; its early finish is going to be 21. Since we have obtained the early start and early finish of all the jobs, the forward pass is now complete and what you can see from this particular forward pass is that the project duration will be 21.

(Refer Slide Time: 14:05)

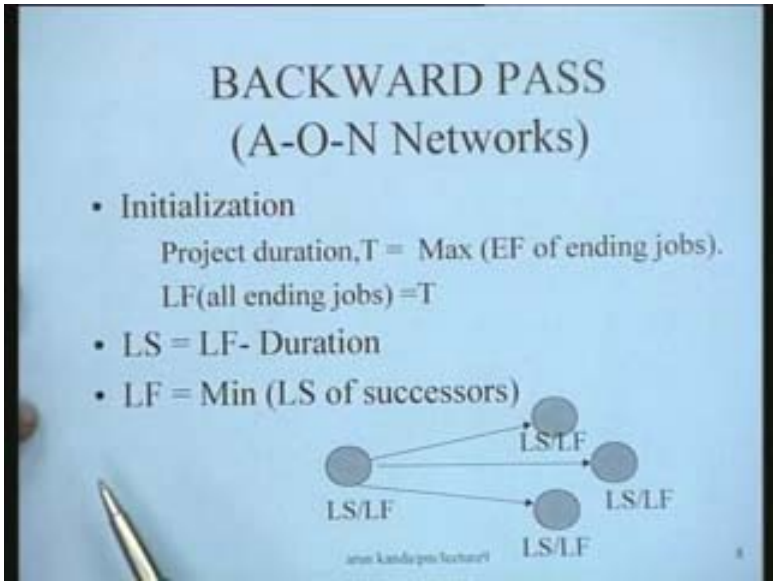



That is the critical path in this particular project will have duration of 21 days. However we would like to do a backward pass and be able to identify the various floats as we have done for the A-O-A network but in this case for the A-O-N network. Let's see how we can do that.

Let's see the mechanism for doing a backward pass. The backward pass also would require an initialization step just as the forward pass required an initialization step and in this initialization what we simply do is that the project duration T is put equal to the maximum of the early finish of all ending jobs. So there could be multiple sinks in a network. One job could be ending at time 15. The other could be ending at 20. Third could be ending at 23. Whichever is the maximum, the maximum of the early finish of all ending jobs will determine the project duration. In the example that we were considering

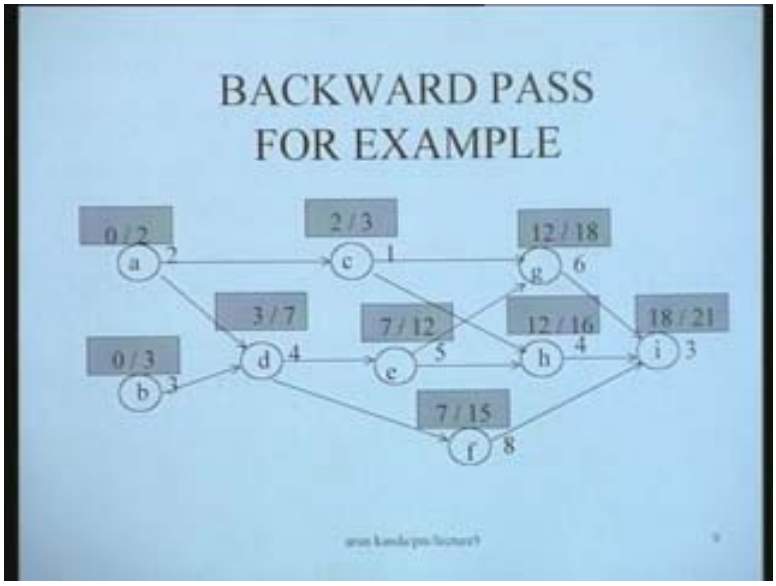
there was only one sink. But in a general situation this is the operation that would have to be performed and then what we do is we initialize by saying that the latest finish of all ending jobs must be equal to T which is the project duration which we have determined in that manner. Then what we do is once we have determined the latest finish of a job, the latest start of a job is simply latest finish minus the duration. We are working backwards and having done that the latest finish of a job, mind you we are writing the latest start and the latest finish at the bottom of the node and we had written the early start and early finish at the top of the node. It is convenient to adopt this convention so that you can identify the numbers that you are computing on the network.

(Refer Slide Time: 16:11)




Here what we do is the late start and late finish has been computed for all the successor nodes of the activity. The latest start is what we are interested in. Look at this latest start for each of the successors and do the minimum and that minimum will become the latest finish of this particular activity and from the latest finish you can determine the latest start simply by subtracting the duration. Now let us try to do this computation for the network. For the network we have already determined the times, the early start and the early finish times corresponding to each of the jobs.

(Refer Slide Time: 16:57)

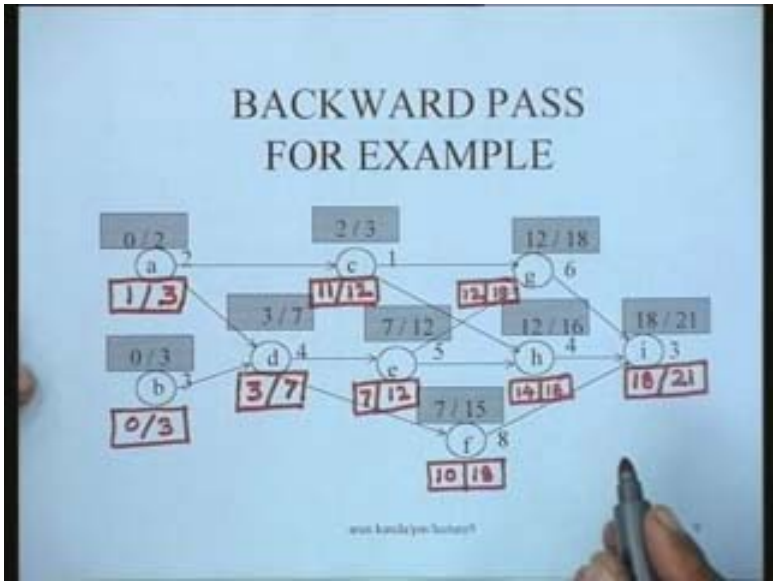



Let's try to do a backward pass on this example. We start with the ending node. The project duration for this particular example is 21. We will say that the latest finish of node i is 21 and 21-3 is 18. The numbers that I am showing you in red reflect nothing but the latest finish and the latest start of the activities. We can put them down like I have shown that here. Then we can come to activity f. When I come to activity f what do I do? This 18 directly gets transferred here and 18-8 would be 10. This is the latest start and the latest finish for activity f. For activity h what would be the corresponding values? There is only one successor for activity h. This is simply going to be 18 and this is going to be 14 because the duration of this particular activity is 4.

Then you come to activity g. Let us see what is the situation for activity g? This is 18. The latest finish for this activity must be 18 and 18-6 will give me 12. Now I have determined the late start and finish for I, g, h and f. Let's try to determine this for activity e. Notice that activity e has 2 successors g and h. What I have to do is I have to take the minimum of 12 and 14 and this is going to be the latest finish for activity e and the latest start for activity e is 12-5 which is going to be 7. This is the latest start and finish for activity e. Let us do the analysis for activity c. Activity c again has 2 successors g and h. The minimum value that I have obtained from these is 12 and 12-1 is 11. The latest start and finish for activity c is going to be 11 and 12 respectively.

Now talk about activity d. Activity d also has 2 successors e and f. The lower value is 7 and the duration is 4. This value is going to be 3 and as far as job a is concerned it has 2 successors again, 11 and 3 being the latest start times. This is going to be 3 and this is going to be 1 and this is going to be the late start and late finish of job a and similarly when you come to activity b this is going to be simply 0 and 3. So we have now completed a backward pass on the network and a forward pass also.

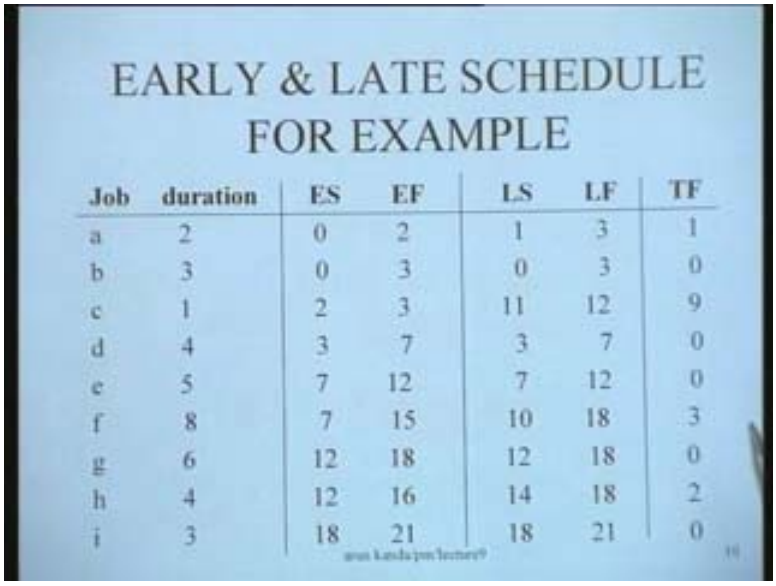
(Refer Slide Time: 20:35)




Incidentally you can see from this that those activities for which the early start is equal to late start, early finish is equal to late finish. Those are the activities which have no float at all and those are the critical activities. The critical activities in this case are b is a critical activity. This is a critical activity. d is a critical activity. Then e is a critical activity. Then g is a critical activity and then i is a critical activity. Other activities are not critical in this. What we can do is we can very easily compute the total float because the total float is nothing but the late start minus the early start or the late finish minus the early finish. This computation can be done.

Activity a has a total float of 1 and activity b has a total float of 0, being a critical activity. From these computations we can actually make a table of floats and we find out that for this particular example activity a has a float of 1, activity c has total float of 9. f has a total float of 3 and h has a total float of 2. The other activities are in fact critical.

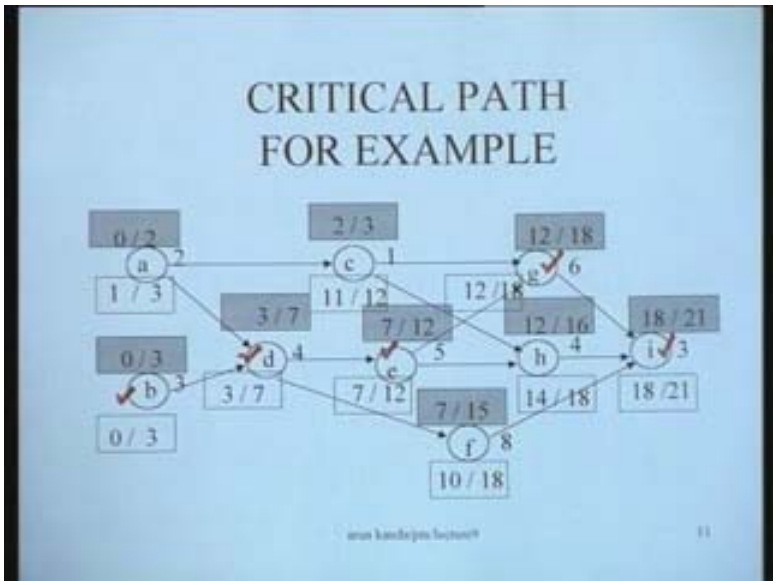
(Refer Slide Time: 21:57)




| Job | duration | ES | EF | LS | LF | TF |
|---|---|---|---|---|---|---|
| a | 2 | 0 | 2 | 1 | 3 | 1 |
| b | 3 | 0 | 3 | 0 | 3 | 0 |
| c | 1 | 2 | 3 | 11 | 12 | 9 |
| d | 4 | 3 | 7 | 3 | 7 | 0 |
| e | 5 | 7 | 12 | 7 | 12 | 0 |
| f | 8 | 7 | 15 | 10 | 18 | 3 |
| g | 6 | 12 | 18 | 12 | 18 | 0 |
| h | 4 | 12 | 16 | 14 | 18 | 2 |
| i | 3 | 18 | 21 | 18 | 21 | 0 |

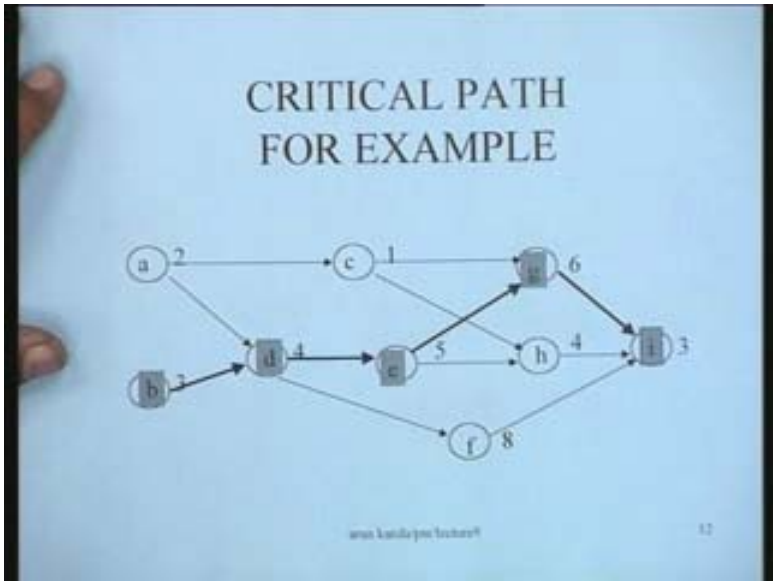
These critical activities are very easy to identify. The critical activities are simply those activities for which the early start and late start are equal. This particular activity b is a critical activity. d is a critical activity. e is a critical activity. g is a critical activity and i is a critical activity. Other activities are not critical because they have certain amount of total float which is greater than zero.

(Refer Slide Time: 22:38)



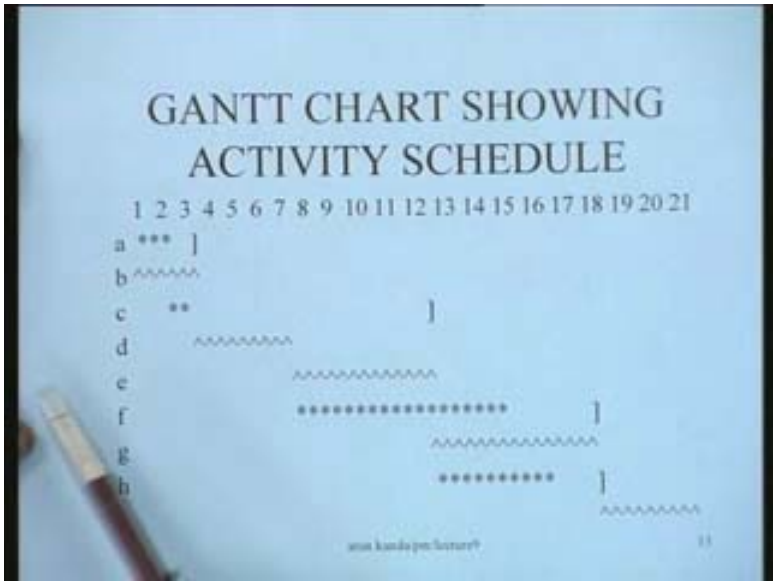

The critical path for this particular example can be shown more clearly here. The critical path for this particular example is b, d, e, g, i and the other non-critical activities are a, c, h and f as shown from this particular representation.

(Refer Slide Time: 23:00)




Since we have already determined the early start and late start schedules for each activity we can represent that information on a Gantt chart and we can get a time table for all the activities and that time table would be shown something like this.

(Refer Slide Time: 23:16)

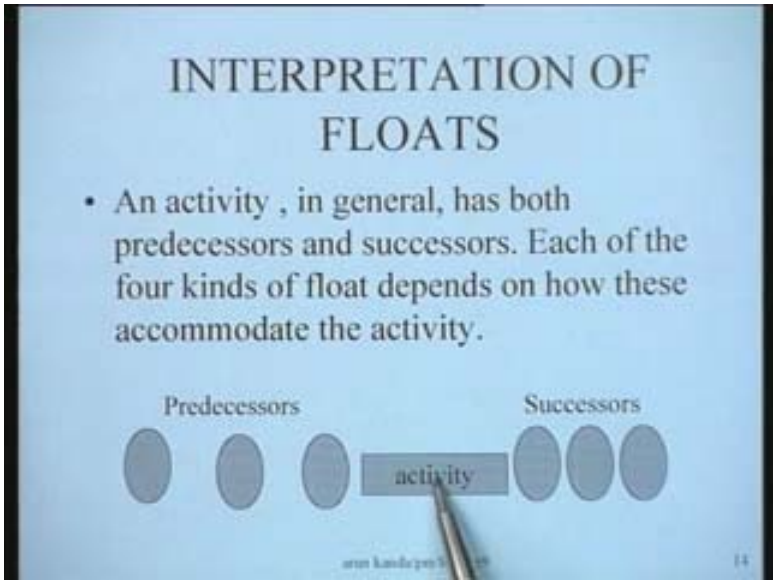



The activities which are critical are shown in this manner. So b is a critical activity and then d is a critical activity and then e is a critical activity and then g is a critical activity and this is a critical activity and they cannot be slided to and fro whereas activities a, c, f and g are non-critical activities and in fact the latest finish of these activities is shown here by means of this. This activity can in fact be slided up to this particular point without

delaying the project. Similarly this activity c which has a total float of 9 can be slided up to this particular point without delaying the project and so on for these activities. This gives you a time table and also shows you the kind of flexibility which is available when you have to implement the project. You know the critical activities in this particular situation. All these durations add up to the critical path duration of 21 days. All the critical activities are adding up to the critical path duration of 21 days. I think that's one of the interesting things. That's an interpretation in fact of the critical path.

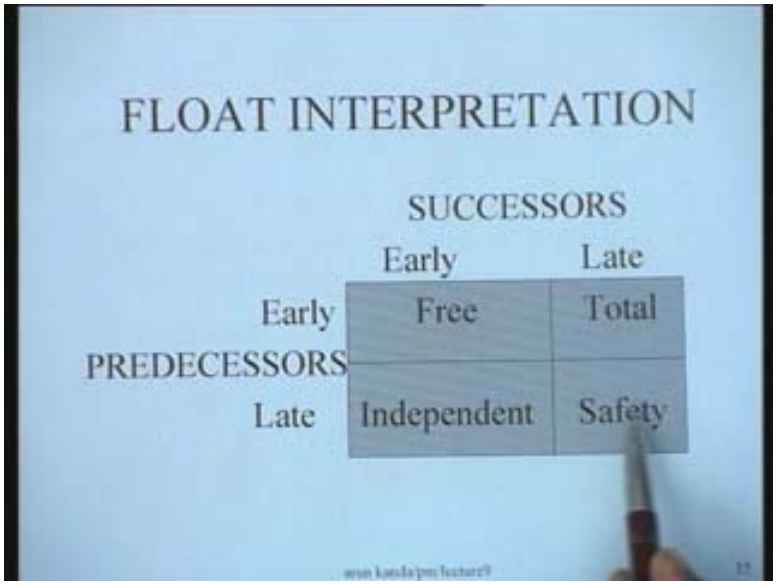
Now we have not been able to compute so far all the four kinds of floats. From the A-O-A representation we showed that float in general depends on how the predecessors and the successors of the activity are accommodating the activity.

(Refer Slide Time: 24:57)



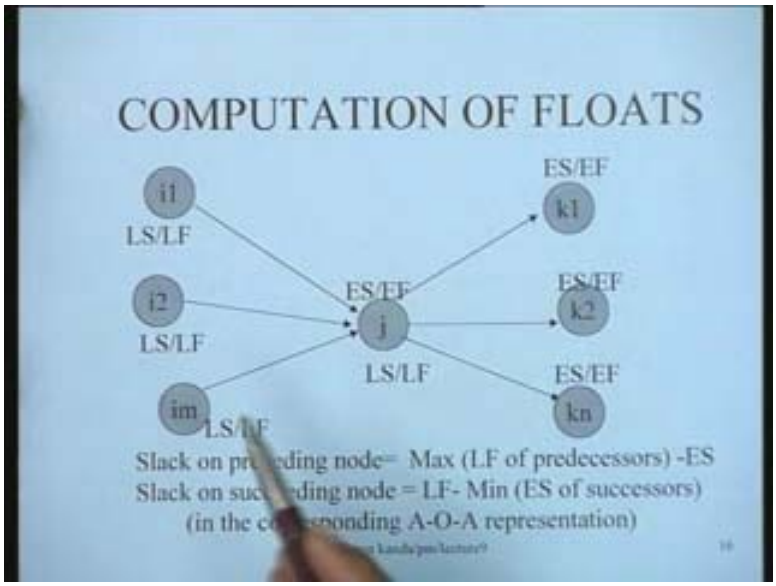

In fact we had talked about 4 kinds of floats and these 4 kinds of floats were if you recall the total float which actually happens when the predecessors are done as early as possible and the successors of the activity are done as late as possible so that the maximum leeway is available to that particular activity. That is the total float and the reverse is the situation with the independent float when the predecessors are delayed as much as possible and the successors wants to get started as early as possible and then you have the independent float if at all it exists in this particular situation and if you have the predecessors being done at the earliest possible and the successors being done at the earliest possible so that you are not eating into the float of the successors then this is called a free float and the safety float has the interpretation when the predecessors get late and you are allowing the successors to get delayed so you have these.

(Refer Slide Time: 25:58)




By using this basic interpretation of the 4 kinds of floats we find that the computation of the total float in the A-O-N network was rather straightforward and we have determined this. We will see how the others can be determined and we are going to basically use this kind of framework. How the computations can be done for the computation of the float here? If we look at a general activity j for which we have computed all these figures basically in the A-O-N network we are computing the early start and the early finish which are obtained from the forward pass. The late start and the late finish which are obtained from the backward pass and we say that we have this information compiled.

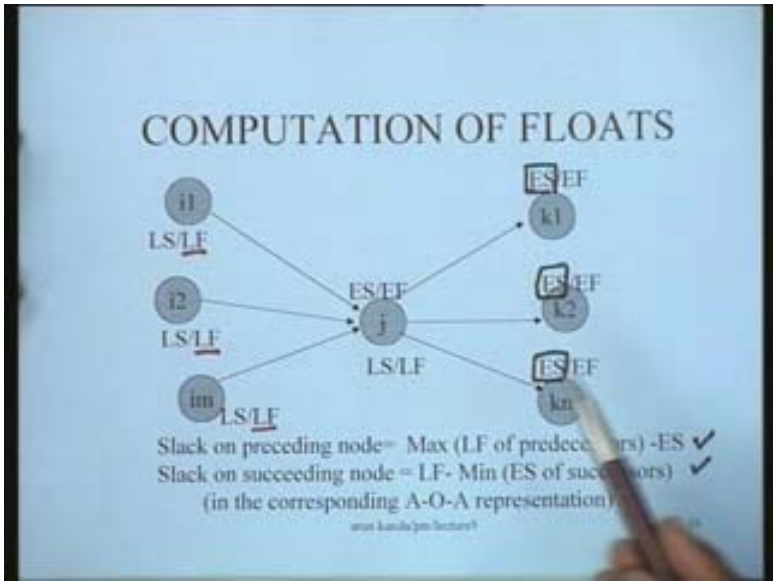
(Refer Slide Time: 26:46)



In the A-O-A network if you have an activity (i j) then the earliest occurrence time of the predecessor node Ei is nothing but ES the earliest start of the activity. That is how you determine Ei. That is rather how you determine the early start of the activity. You calculate the early start of the activity by putting it equal to Ei. This is nothing but Ei and by the same logic LF is nothing but Lj of the corresponding A-O-A network. We have these two. This is the predecessor node earliest occurrence time. This is the successor node latest occurrence time in the corresponding A-O-A network and to compute the floats what we need to actually compute is the node slacks. That is we need to compute the Li minus Ei. That is the slack on the predecessor node and the slack on the successor node. How we can do that is very simple. In this case we say slack on the preceding node in the corresponding A-O-A representation is nothing but, we have this, all the predecessor jobs for this particular activity. We will take the maximum of the late finish of jobs.

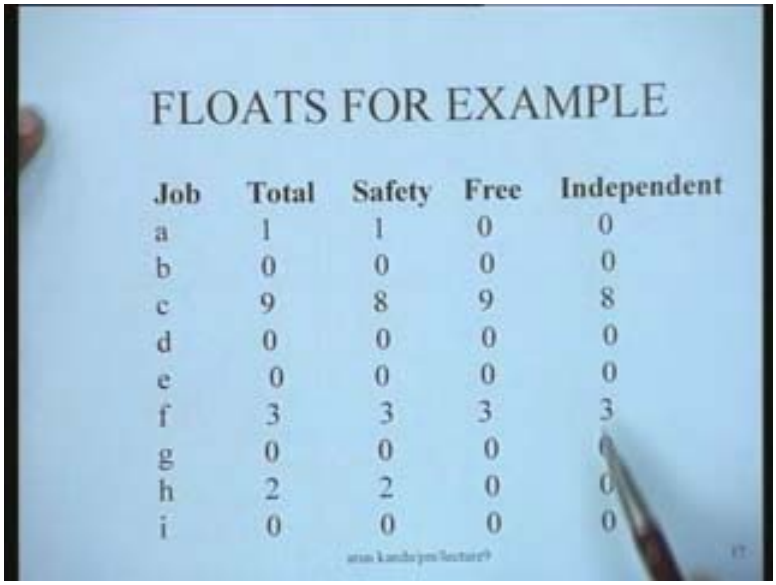
The numbers which are shown at the bottom of the activity of the predecessors, we look at all these numbers and what we try to do is simply we try to look at this number, we try to look at this number and we try to look at all these numbers the latest finish of all the jobs and we try to take the maximum of these numbers. The maximum of the latest finish of all predecessors would have the interpretation of Li on the A-O-A network and ES is nothing but your Ei. Li minus Ei would be in fact the slack on the preceding node which we are looking for. This is the point. Actually for all the predecessors you would need the LF value of the predecessors. You can perform these computations. Similarly if we have to find out the slack on the succeeding node, we have Lj which is LF; we need, sorry, we have Lj, we need Ej. The earliest occurrence time would be nothing but the earliest start of the successors. So you look at these successors and you find out the first value here; this particular value here. In fact we look at this particular value, we look at this particular value and we look at this particular value here and what we do is all these values we find out the minimum of these and the minimum of these. Minimum of the ES of successors would in fact give me nothing but the Ej value in the corresponding A-O-A representation. I have a means of computing the slack on the preceding node by using this particular formula and the slack on the succeeding node by using this particular formula and mind you this particular representation comes from that basic interpretation of the various kinds of floats that we just try to see.

(Refer Slide Time: 30:23)




What we can then do is by using this interpretation we can in fact compute all the floats for the example and this computation for all the floats, for the total float, the safety float, the free float and the independent float for the entire example is shown here.

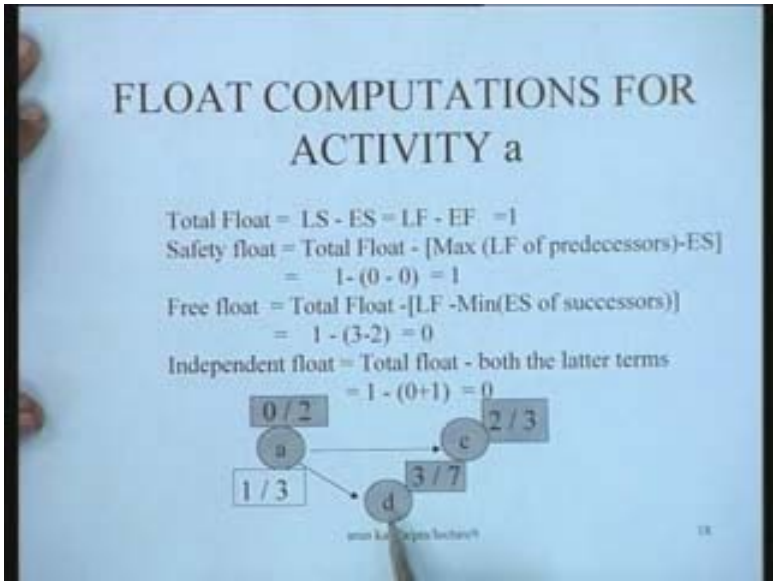
(Refer Slide Time: 30:44)




Notice that for activities which were critical all the 4 kinds of floats are zero. So b, d, e, g and i these were critical activities. For them actually we don't have to actually compute all the floats. They would be zero. However for each non-critical activity we can compute the safety, free and independent floats in the manner that I just indicated to you. To show you how these computations are performed let's take activity a in the network. I have

isolated the activity a and notice that activity a is a beginning activity. It has no predecessors. It has two successors and these successors are c and d. What I have noted down here is the early start values, early start and early finish of successors. That's all I need for the activity.

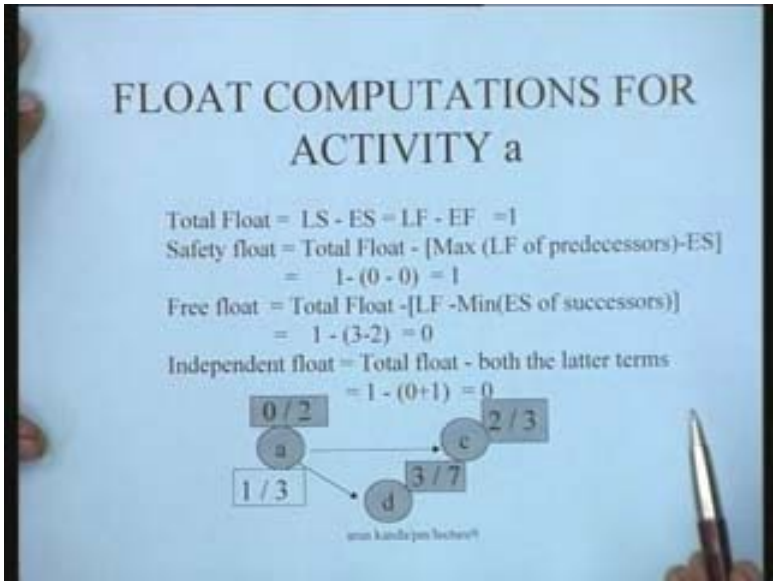
(Refer Slide Time: 31:47)




The total float for this activity is nothing but latest start. Latest start is 1 minus early start or latest finish minus early finish which is this particular value. This is 1, the total float. The safety float for this particular example will be the total float, the total float we have already computed, minus the slack on the preceding node. This is maximum of LF of predecessors minus ES. When you take the predecessors in this particular case it is a beginning activity. It has no predecessors. This particular value is zero and early start for this activity is zero. The safety float for this activity is 1.

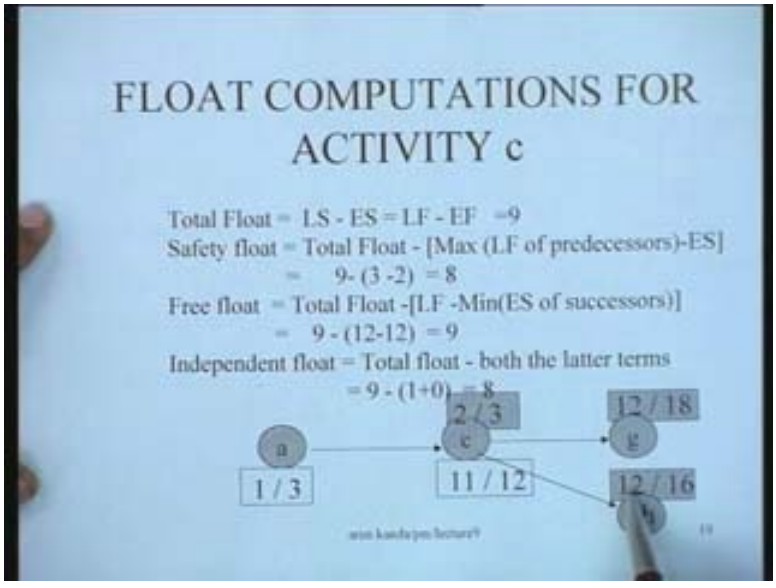
Let us now compute the free float for this activity. The total float minus the node slack on the successors in the corresponding A-O-A representation is given by LF. The latest finish for this activity is simply 3. This we have computed already minus the minimum of the early start of successors. Only for the successors I have the early start information. So minimum of 2 and 3 is 2. So this is 2. 1- (3-2) is 0. The free float for this activity is 0. Similarly the independent float is nothing but total float minus both the latter terms that is this term and this term both 0 and 1.1-(0+1); this is zero.

(Refer Slide Time: 33:21)




We have computed all the floats for this particular activity. For clarity or for further demonstration we can look at each of the non-critical activities and show you how the float is computed. Let us see for instance activity c. When you see activity c this is an activity which has both predecessors and successors. This is the activity; early start early finish, late start late finish. For the predecessors I have got the late start and late finish information. I don't need the other information. So I have only taken the relevant information and for the successors I need only the early start early finish representation.

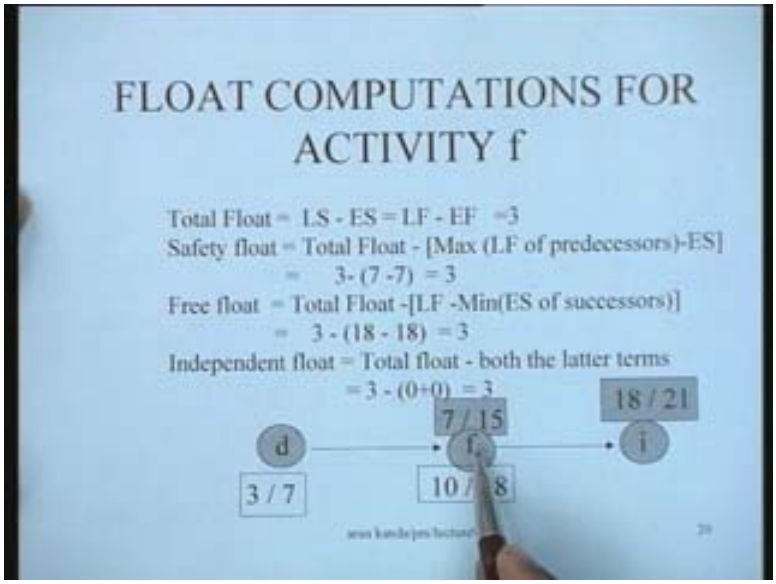
(Refer Slide Time: 33:59)



The total float for this activity is simply 11-2 which is 9 and the safety float is nothing but 9 which is the total float minus the maximum of the latest finish of predecessors. maximum of the latest finish of predecessor There is only predecessor here and that predecessor has a maximum latest finish time of 3. So 3 minus the early start of the activity which is 2. The safety float for this activity is 8 by this logic and similarly when you are computing the free float for this particular activity this will be total float minus late finish minus minimum of the early start of successors.

How are you going to compute the early start of successors? These are the two successors. The minimum of these two values happens to be 12. The latest finish of this activity happens to be 12. So 12-12 is 0; 9-0 is 9. This is 0 and this is 1. So 1+0 you subtract from 9 and you get the independent float which is 8. So we have 9, 8, 9, 8 for activity c. In fact something similar could be attempted for activity f. Activity f was a unique activity which had only 1 predecessor and 1 successor.

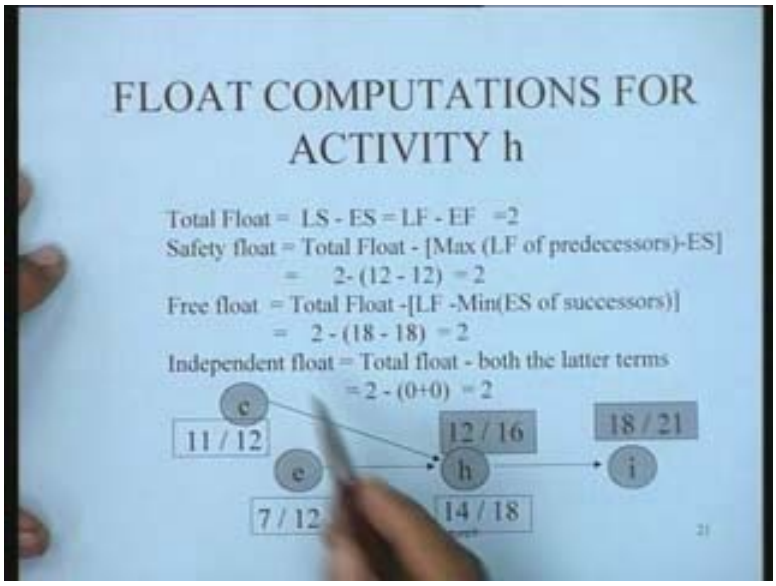
(Refer Slide Time: 35:23)




The minimand and the maximand would be a very trivial thing for this activity. We are having here the total float which is the maximum of the late finish of predecessors which will be 7; 7-7. The early start of this activity is 7 and the free float will be total float minus latest finish. The latest finish of this activity is 18 minus the minimum of the early start of successors and there is only 1 successor. This is 18, this is 3. All the 4 floats in this particular situation come out to be equal. They are in this particular situation as shown here. Similarly for activity h, h had 2 predecessors and 1 successor. Knowing the information which was available from the forward pass we could in a similar manner compute all the information for this particular activity h in the manner that we computed here and here again as you notice for activity h the latest start is 14, the early start is 12. The total float is 2 and when you are computing the safety float you have to take from the total float, the maximum of the latest finish of predecessors. Latest finish of predecessors

is 12 and 12. You have to take the maximum of that which is 12 minus the early start of the activity which is 12 itself. So you get this particular value.

(Refer Slide Time: 36:58)




Similarly the free float is the total float which is 2 minus the late finish of the activity. The late finish of the activity is 18; 18 minus minimum of the early start of successors. There is only 1 successor so the early start of that successor is 18. This is 2 and since these values are coming out 0, the independent float will actually be equal to 2. In this case also all the 4 floats are coming out equal. You would recall that h was an activity in the corresponding A-O-A network which had a dummy preceding and this was an activity for which we had discovered anomalies in the computations of the 4 floats. Here there are no such complications and in this case we get all the four floats as equal for activity h.

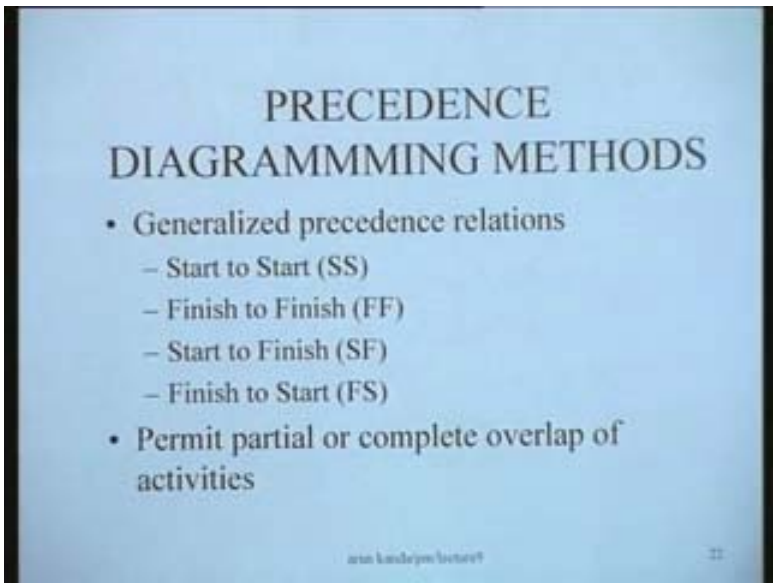
Having talked about the basic scheduling in A-O-N networks, we are going to be talking about a very important class of specialization of A-O-N networks known as precedence diagramming methods. Precedence diagramming is essentially an activity on node relationship but we have generalized precedence relationships. The only kind of precedence relationships that is possible in a conventional A-O-N network or in A-O-A network is the finish to start relationship. That is an activity must be finished before the successor activity can start or a set of activities must be finished before the next activity can take place. What is been argued is that there could be more relations than just this relationship and in fact there is a possibility that a number of activities might take place concurrently. You want a number of activities to take place concurrently but not totally concurrently but you might want to build in some constraints.

Let’s see what kinds of constraints can be built in and how do we do the programming here? Four kinds of lead lag relationships are possible. You can constrain either the start to start of an activity or the finish to finish of the activity or the start to finish of the

activity or the finish to start of the activity. There are only 2 possibilities. You are constraining in start to start after an activity starts. Some time after that, minimum time after that you might want to start the next activity but you might not weight for the finish of that activity. Imagine for instance that you are plastering a large piece of road and what you would do is that you would probably like to lay all the concrete first and then do the plastering once the concrete is over. You need to necessarily wait for the concrete to set. If it's a long wall you can say that the first portion once it is leveled you can start doing the plastering activity on that portion itself and then continue with that activity subsequently. Then if you want to impose a restriction there you can use an SS lag. If I specify an SS lag it would say that the start of the plastering can start after the start of the pouring of the concrete after may be 5 hours, if I specify a time of 5 hours and most of these activities can take place in parallel.

Something similar is possible with a finish to finish lag or a start to finish lag. In fact the conventional networks have only a finish to start lag with an FS value equal to 0. You finish something and then you start. The major advantage here is that you are permitting partial or complete overlap of activities. That's the major advantage in precedence diagramming methods.

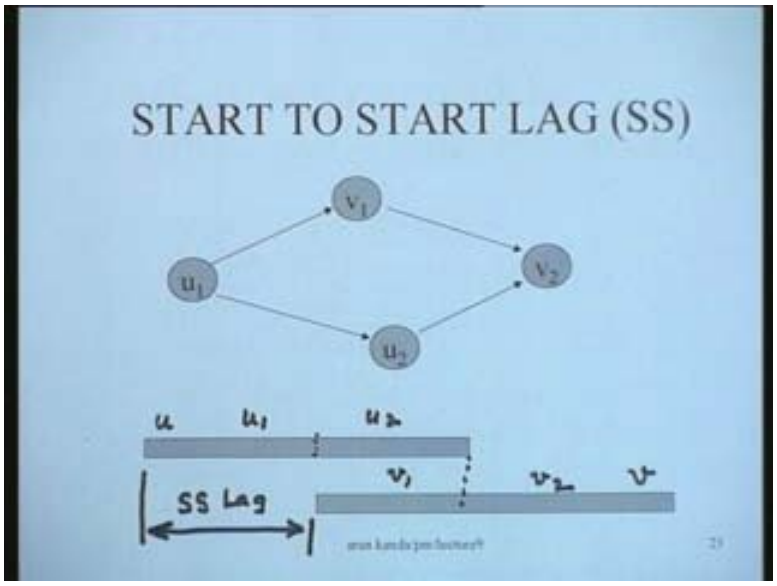
(Refer Slide Time: 41:27)




Let's look at the 4 types of lags before we take an example. For instance if you have a start to start lag what happens is we have 1 activity. Let's call this activity as u and we have another activity. Let's call this activity as v and we would like to retain a partial overlap between these. If we were doing the conventional situation then one activity will start only when this is finished. A start to start lag means essentially that this particular interval, which is called the SS lag, is specified. By specifying a certain lag, if this is specified as 4 days what it means is that the start of activity v will take place at least 4 days after the start of u. Its constraint is only on the start. This is what we mean by an SS lag. Actually this can be represented as a conventional A-O-N network because what you

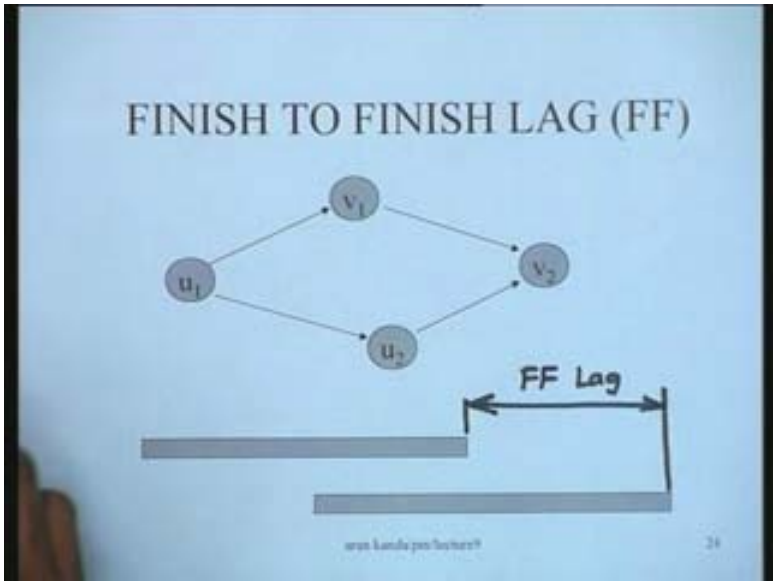
can do is you can split up activity u into $u_1$ and $u_2$ and activity v into $v_1$ and $v_2$ and these are in series. So $u_1$ and $u_2$ is the activity u and $v_1$ and $v_2$ is the second activity v. This is $u_1$ which is the non common portion. These 2 activities are in parallel and this is the activity which is subsequently alone after this. It is to accommodate this kind of parallelism that we can use an SS lag.

(Refer Slide Time: 43:06)



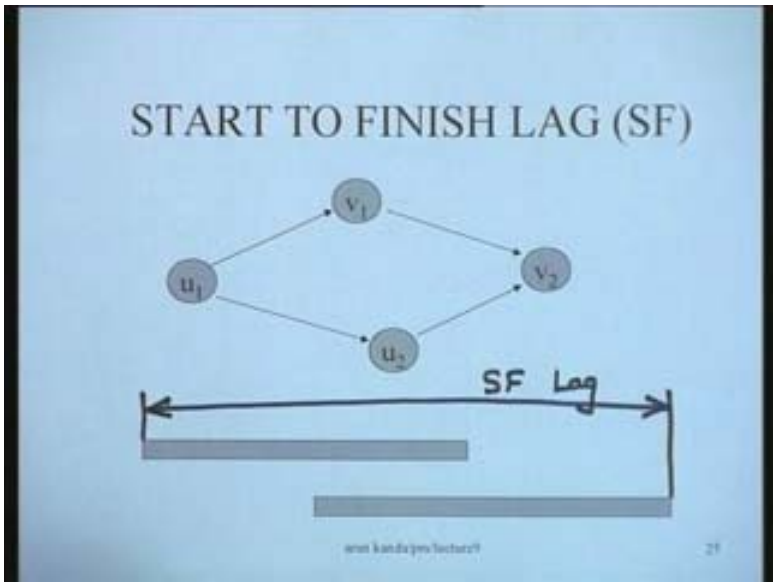

Let us now see what would be the implication of a FF lag. Again if we were constraining or specifying an FF lag it would mean that we are saying that from the finish of this activity to the finish of this activity we are specifying a minimum amount of time which is the FF lag and this FF lag we will say that from the finish of this to the finish of this there should be at least this much of difference and it can have the same kind of interpretation for the project.

(Refer Slide Time: 43:40)




We can have a start to finish lag. A start to finish lag would mean from the start of this activity to the finish of the second activity and this particular duration is what we call the start to finish lag. You could specify a value for this particular start lag to finish lag. Knowing this you can compute what time this can finish and from there you can then compute what time this can start at the earliest.

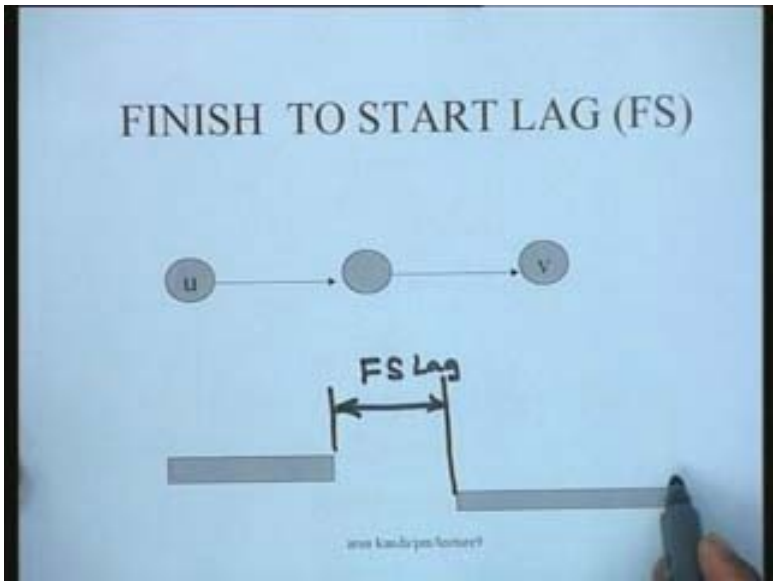
(Refer Slide Time: 44:17)




This is the SF lag and we can have the finish to start lag. That is what we conventionally use in A-O-N networks. From the finish of this activity to the start of this activity this

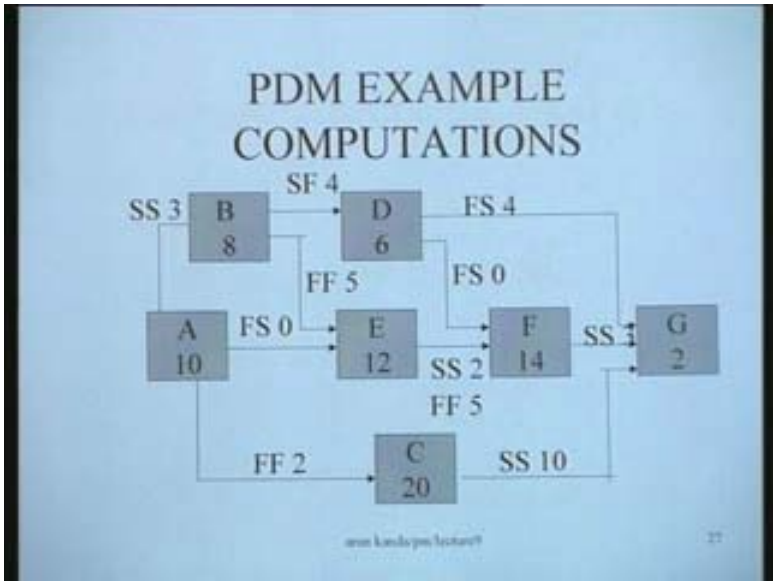
particular interval of time could be specified as a finish to start lag. That means it's like trying to say that rather than introduce a separate activity like you can do here for this ……. thing you finish this activity. But you would like this to be cooled for a certain amount of time before you can do further operations on this. You can specify a finish to start lag for the activity.

(Refer Slide Time: 45:05)




What we have specified is that these are the 4 kinds of lags which can be used and we can perform in fact basic computation on the project network by using these kinds of computation. Let's take an example of a project network which is a PDM network. In this particular network we have the activities which are shown here A, B, D, E, C, F, G, etc and the various kinds of lead lag factors are shown here like between A to B it's an SS lag specifying a value of 3. Between B to D it is an SF lag of value 4. Between B to E it's an FF lag of value 5 and so on. Let us try to do a basic forward pass and a backward pass on this particular network and see what the results would be. The logic is very simple. The only thing is that for each activity you would try to find out in a forward pass the early start by the various constraints and the most constraining constraint will actually determine the forward pass.

(Refer Slide Time: 46:22)



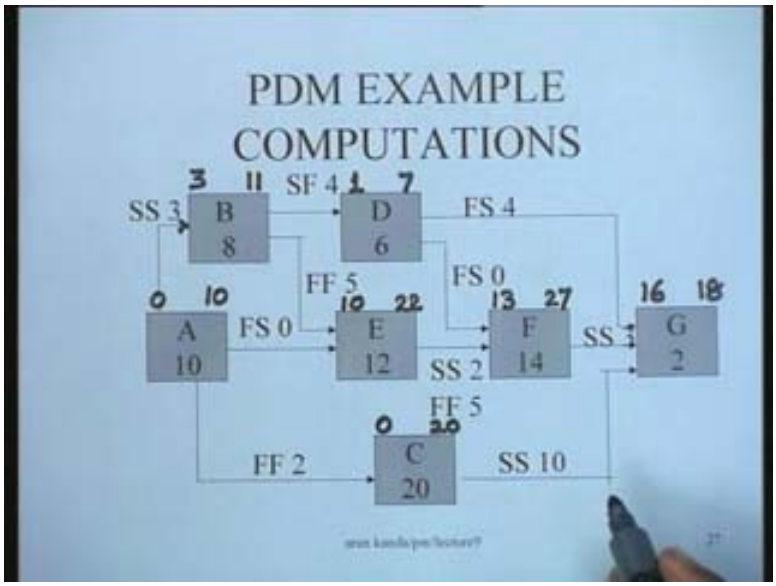

For instance let us start the computations. Let's try to determine the early start. I will put the early start. A is the starting activity. Early start for this activity is 0. I am putting them on the left hand corner of each box. This is the early start of the activity. The early finish of this activity is going to be 0+10 is 10. Early start and early finish for this particular activity is 10. Now notice something interesting. Let's look at activity C to begin with. Activity C has a finish to finish lag with regard to activity A which means that 10 is the finish of this particular activity. From the finish of this, 10+2 that means this particular activity must finish not earlier than 12 days. It has duration of 20. It will start much earlier than this. So what we can say is it will be later but since the project starts, so we can say that the earliest start for this particular activity will be 0 and if the earliest start is 0 this particular activity will finish at 20 and that does not violate this constraint because this constraint says that it must finish later than time 12. It finishes now at time 20 by virtue of the fact that the project starts at time 0. We have determined the early start and early finish for job C.

Let's look at job B. This is constrained here by a SS relationship. The earliest start of B by this will be 0+3. We can say that B can start earliest at time 3 and if it starts at time 3 it can finish at time 11. Now let's take job E for instance. Job E has two kinds of constraints which are coming into it. We can see both. If we come from here, it says finish to start has a time of zero. The earliest that this can start by virtue of this particular constraint is 10 whereas if you see here this is finishing at time 11 and it says finish to finish. This must finish at time 11+5 which is 16; 16-12 is 4. This is 4 and the earliest start that you get from this particular relation is 10. The maximum of the two is 10. So this earliest start is 10. We have considered this FF constraint. We find that this between this and this, this is the dominant constraint and this particular duration is 12. So this particular activity will have a time of 22. That means if it starts at time 10 it will have a finish time of 22.

Take activity D. Here we have a start to finish lag. Start to finish lag means that it starts at 3; from the start of this to the finish of this. We are saying that this should take place not earlier than time 7. This means when it takes place at time 7, it has a duration of 6. It must start at 1 and it must finish at 7. We can do this way. Now you are noting something very interesting. This is an activity which is shown following this and yet in terms of the schedule it will be in parallel and it will start earlier than this and will in fact finish earlier than this activity. This is a feature of PDM networks. This can happen.

Now let's take activity F. When you take activity F what is the earliest start of this particular activity? The earliest start of this particular activity we work out from here. This finishes at 7 and finish to start lag is 0. This means from here we can start earliest at 7. But here 2 constraints are specified. There is an SS lag of 2. SS lag of 2 means it can start earliest at 12. That's one thing and at the same time there is an FF lag of 5. When there is an FF lag it means it is finish to finish lag; 22+5 is 27. 27-14 is 13. Out of all the three, 13 is the largest value. The earliest start of this particular activity is 13. We have computed from all these, this was 7. This was 12 and from all these three possibilities this is 13. This project will finish at time 27. Here SS is equal to 3. There are three constraints coming on activity G. As far as activity G is concerned this is finish to start lag. That means the start of this must be at least 11 from this particular thing and this says 13+3 it must be 16 which is higher and from this particular stage it is 10. Which is the highest of all the three? So the start time for this will be 16 and then the duration is 2 and then this is 18. We have now completed the forward pass.

(Refer Slide Time: 52:18)

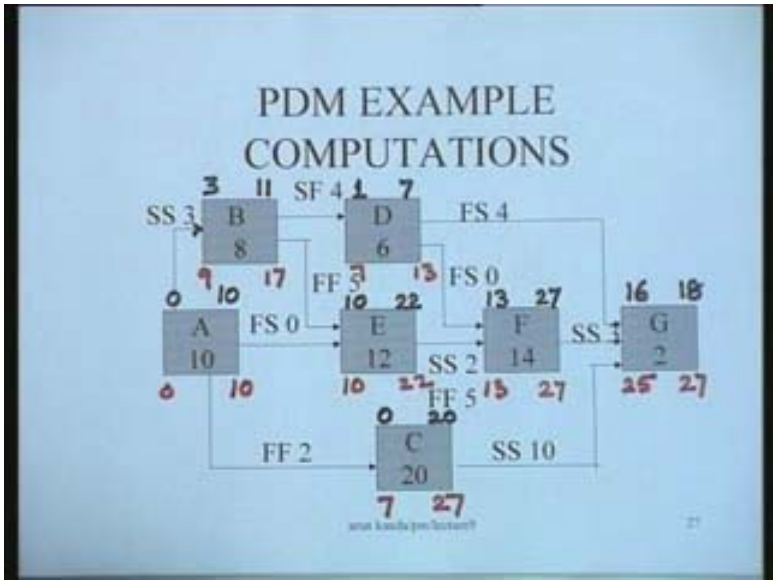



What is the project duration? 20. Here is an interesting thing. You find that the project duration for this particular example is the maximum of these times and it is 27 and it is occurring for activity F and yet there is an activity G which is succeeding this activity. But it has other kinds of lead lag relationships and in fact it concludes much earlier than

this particular activity. This is the basic idea of performing a forward pass for this particular example.

We can also do a backward pass by a similar logic. You have to consider all the constraints but for initialization these are the two terminal activities. So you will put 27 here and 27 here. That is the latest finish time of all these activities will be taken as 27. If we put 27 here, 27 here we have this and if you continue this process you find that the late finish time of this is 27 and the late start time of this particular activity is 7 and similarly for this particular activity this is going to be 13 and let us say for activity D this is going to be 13 here and its going to be 7 here and then if you compute for activity E it will be 22 here and 10 here and if you compute for activity B it will be 17 here and 9 here and finally for activity A you would get 0 and 10. Exactly in the same manner we are working backwards and for 27 this would be 27-2 which will be 25. The numbers in red here show the latest start latest finish and the numbers in black show the early start and the early finish for various activities. I have assumed in this example that all the activities cannot be split.

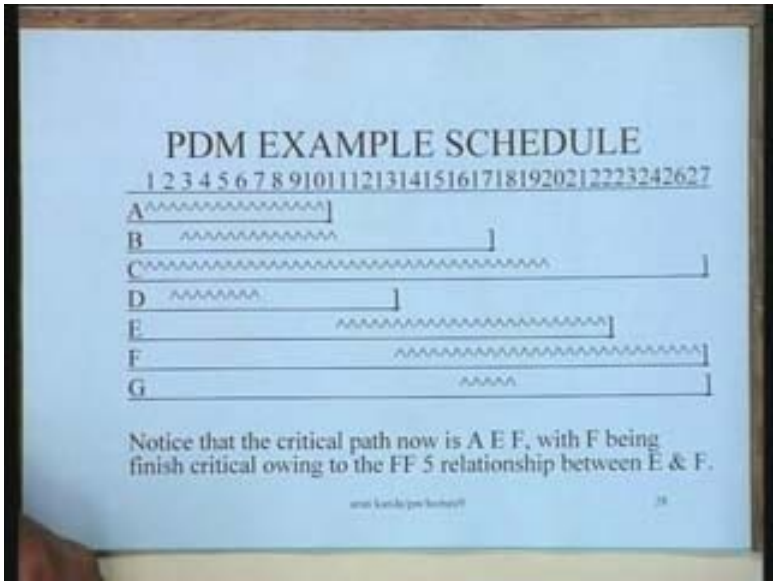
(Refer Slide Time: 54:20)




But there is a possibility that activities may be allowed to split and in that case you can obtain different kind of situation. Here which are the three critical activities? The three critical activities in this case are A, E and F and yet the addition of their durations does not give you 27. That's another interesting thing in this example.

If you draw a Gantt chart for the problem what we find is that in this particular case the critical path is A, E and F. A, then E; it goes like this and then F. Because there is a finish to finish relationship between this which determines this, so actually F is finish critical. A and B added up and then part of this is in parallel. The total duration is 27 but they do not add up to the project duration as was the case in the conventional A-O-N networks.

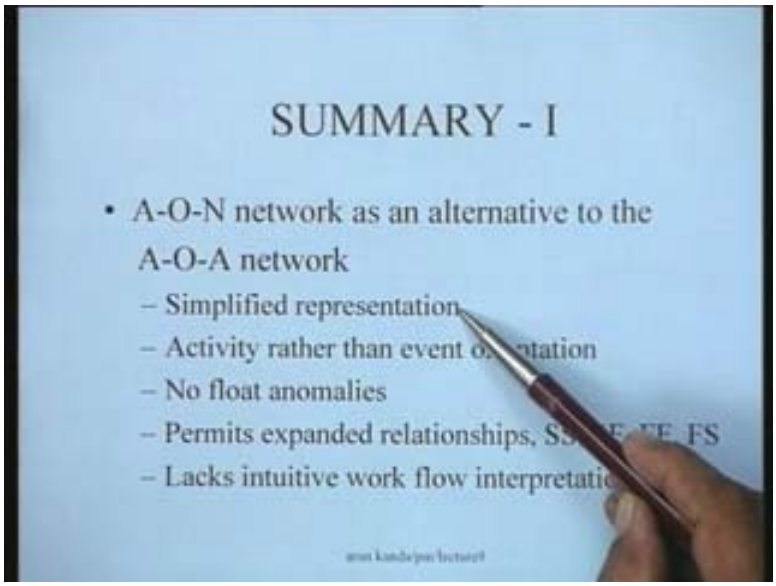
(Refer Slide Time: 55:22)




I think this is one feature which is common to all precedence diagramming that is because of the overlap of activities.

In this particular lecture we have looked at the A-O-N network as an alternative to the A-O-A network and we have found that the A-O-N network gives us a simplified representation.

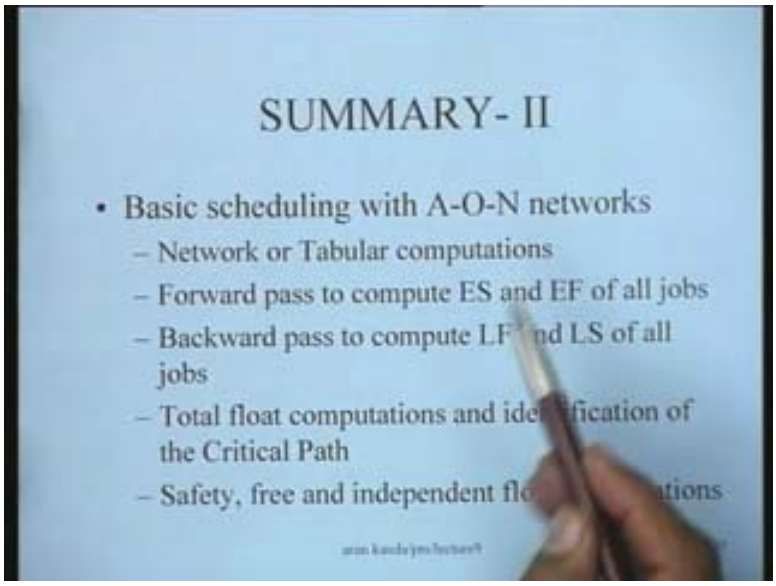
(Refer Slide Time: 55:45)




It has an activity rather than an event orientation. There are not float anomalies. It permits expanded relationships of the type SS, SF, FF, FS. However it lacks intuitive work flow

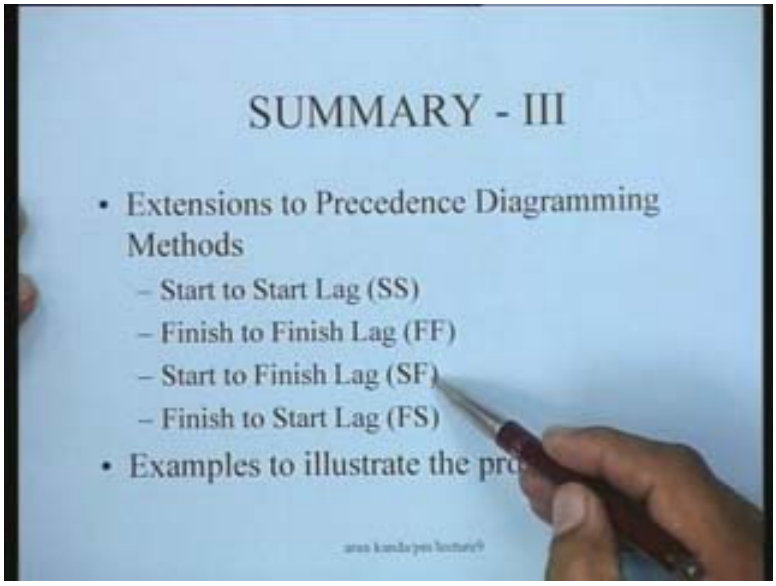
interpretation. That's one of the things that we have seen. Then we have done basic scheduling with A-O-N networks. We can see that the computations can be performed on the network or equally comfortably in a table if you retain for instance in a forward pass the set of predecessors and the successors of the activity. Then you can calculate in a tabular form.

(Refer Slide Time: 56:25)




We have seen how to do a forward pass, a backward pass, total float computations and the identification of the critical path and the computations of the safety, free and independent floats directly from the whole situation and finally we have seen some extensions to precedence diagramming methods where we have considered these four types of lags.

(Refer Slide Time: 56:48)




The start to start lag, the finish to finish lag, the start to finish lag and the finish to start lag and we have taken examples to illustrate these procedures. Thank you!