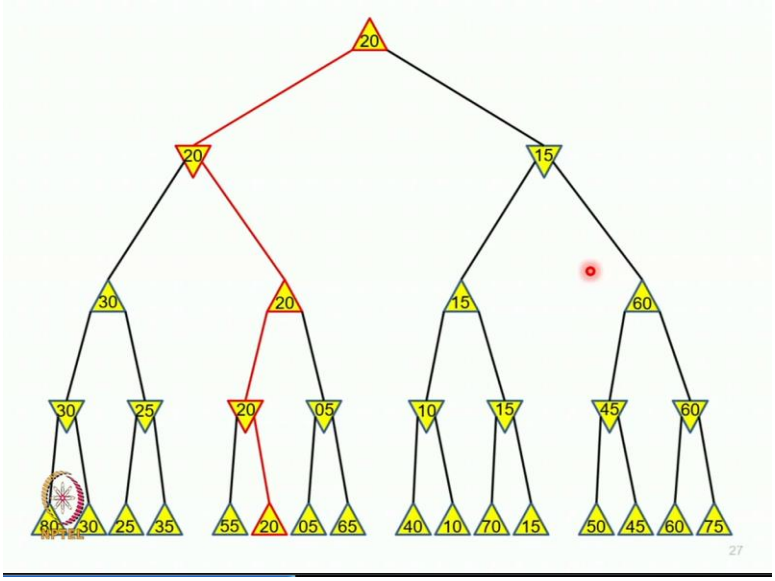
Artificial Intelligence
Prof. Mausam
Department of Computer Science and Engineering
Indian Institute of Technology Delhi

Lecture-31
Adversarial Search: Alpha Beta Pruning-Part-3

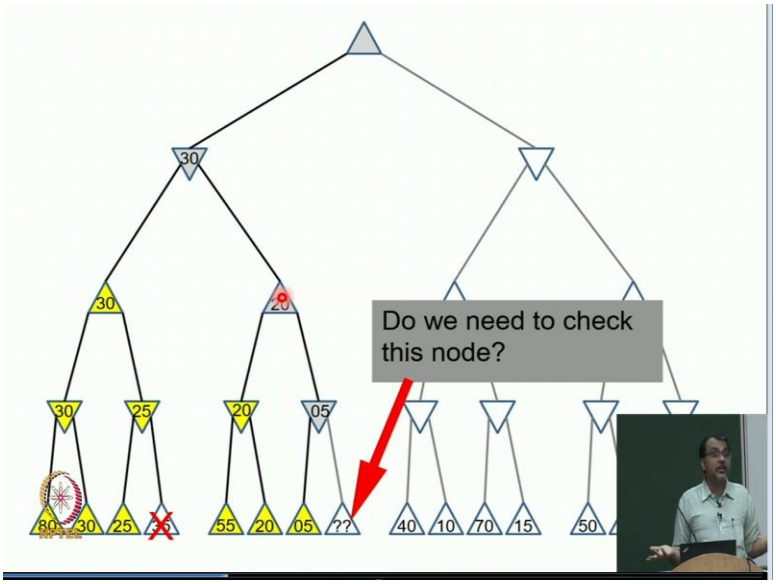
We are going to learn what is called alpha beta pruning and go back to your idea of depth first search branch bound. We are trying to figure out what does bounding mean in the context of mini max. Now, I told you look at this tree and tell me if there is a node we should even not be we should not even be evaluating.

(Refer Slide Time: 00:45)



Is there any node you think is wasteful? We should not have bothered to evaluate it. Did somebody figure such a no doubt? Yes, the person in the white and black what is your name? White and black yes Naman 35, let us look at 35 why do you think 35 does not matter. So let us look at this here.

(Refer Slide Time: 01:40)

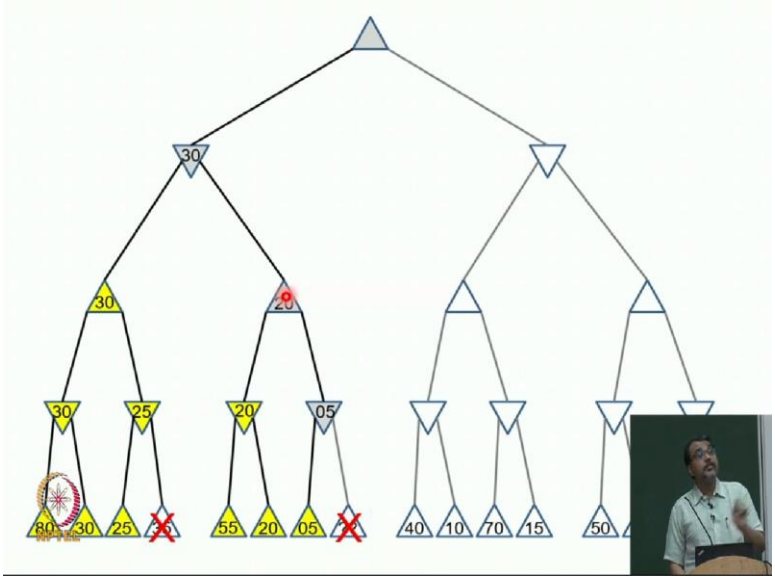


Let us say we are at this point I have evaluated the min 30. I have sent this information to the max it now knows that my left sub child can get 30 it goes right. The min gets to know about 35. We are at this specific point in the junction the question we need to ask is do we need to check this node? Does it matter what value it is? Let us say this value what can it be, it can be less than 25 or greater than 25 it could be greater than 30 for all we know, but let us say it is greater than 30 then what is the parent going to do?.

Go left, it may goes left the value would be come on this is important. I really need all your concentration. So, if this value ends up being let us say very high 1000 what is the min node going to say? Min node is going to say 25 and it will say I do not care. I will not take 1000 it is going to make my adversary even the human, machine win I do not want the machine to win and if it this remains 25.

What is this parent the max going to do? 30 on the other hand, if this question mark is let us say minus 1000, what is the min going to say minus 1000, but what is the maximum to say 30? So, irrespective of what value this question mark is, I know that I will never come here in the optimal play. The reason I will never come here is the max already has a solution of 30 and I have a solution which is less than equal to 25 and because I have a solution which is less than equal to 25 my, the max is never going to come here therefore.

(Refer Slide Time: 04:00)

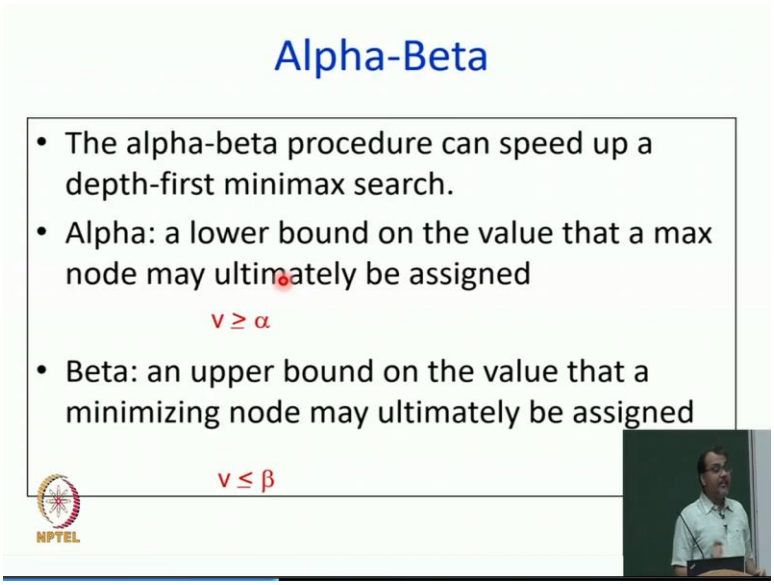


This branch is now guaranteed to be worse than what max already has. And therefore, we can prune away trying to even evaluate this node. Let us take one other example. Can you see one similar example? That comes later 65. Very good. Let us look at the 65.This max already has a solution of 20. And the min has a solution of 5 or less. The min has solution of 5 or less, and the max has solution of 20 or more. See, it is a max. It already knows 20. So now its value can only increase it cannot decrease because it is a max.

So max has a solution of 20 or more min has a solution of 5 or less, they can never reconcile. And therefore there is no point in evaluating this. And while this intuition hopefully makes sense to you, we need to figure out how to operationalize it as an algorithm. And that is why I said thing depth first search branch bound in what is the bounding procedure in the bonding procedure, suppose you are maximizing the bounding procedure basically says, that I have found a solution and now my best solution is going to be this solution or better.

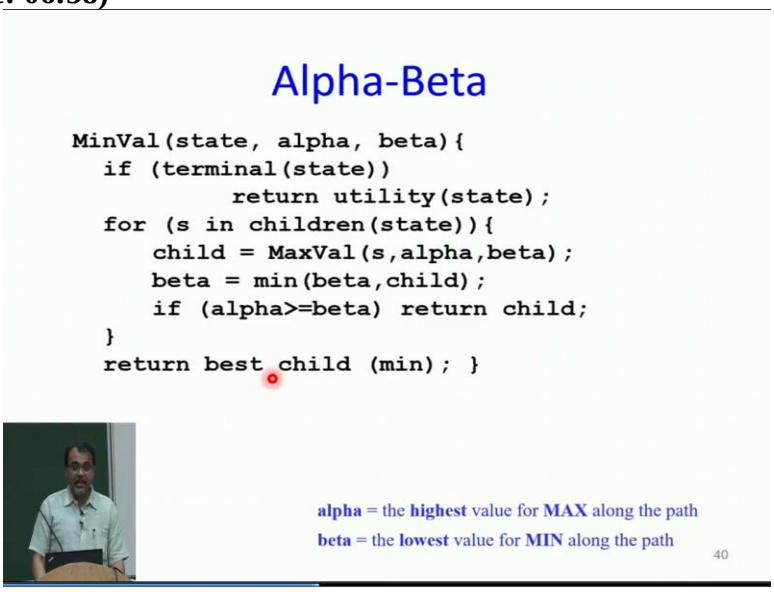
That is my bounding procedures and whenever I can prune that my best solution is going to be worse than I can prune it out. And the same intuition we are going to do except that we will not just have best and better, we will also have worst and worst and then it will compare the 2 and this particular method will be alpha beta pruning.

(Refer Slide Time: 06:02)



So, alpha beta procedure can speed up a depth first mini max search alpha is a lower bound on the value that a max node may ultimately be assigned, it is like the value of the max is 20 or higher. So, the 20 would be alpha and beta would be an upper bound on the value that a minimizing node may be assigned like I said value is 5 or lower. So beta would be less than equal to 5, beta would be 5. And whenever alpha is greater than beta, I can prune out because I know that max will do this or more min will do this or less. So therefore min is no longer or they are no longer compatible, and so I can prune it up.

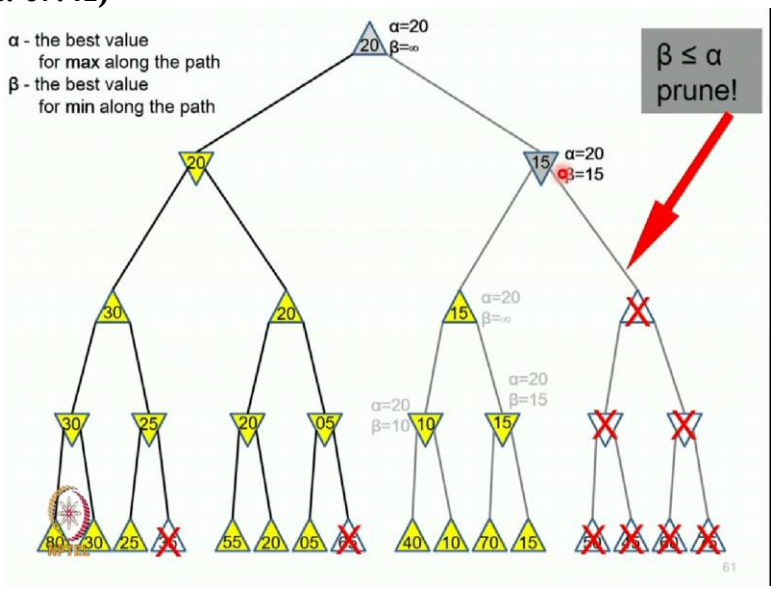
(Refer Slide Time: 06:58)



So this is the main change in the alpha beta procedure. So, this is the mini max algorithm. But now, what I have said with every child in the min I will maintain the beta which is the min of the

beta child. And if alpha is going to be greater an equal to beta, I am going to prune and just return the child. So, let us look at it in the context of an example and after this example will continue this next class. So, I want everybody's concentration for the next 5, 7 minutes, this is important once will figure this out. You are done I mean, this is the most interesting part of this game playing chapter.

(Refer Slide Time: 07:41)



So, theoretically that is, so initially I know nothing. So, initially my alpha is the worst possible what is the worst lower bound minus infinity. So alpha is minus infinity. Similarly, my beta is the worst possible it is the worst upper bound. So it is infinity. And initially, alpha is not greater than equal to beta, so everything is fine. Now, as soon as I get 80, I send this information up. But not only do I send this information up, I also change one of alpha beta for this node, what do I change? This is important.

It is a min node. So min node is saying my value will either be 80 or lower. So I am going to change my upper bound, and what is the upper bound? That is beta. So now my beta became 80. Similarly, I get to 30. When I get a new value, my beta becomes 30 and when I update this information up send, back up to this max node. Now my max node knows that its value is going to be 30 or higher. So, I am going to update my lower bound which is alpha.

So at this point, what I have done, I have said that I know a path, my value is going to be 30 or higher or the best value is going to be 30 or higher. Now, the interesting thing I am going to do is

that when I go down, I will send this information down. So, when I go down now, my next node also says that its alpha is 30 and its beta is minus infinity or beta is infinity, because it is maintaining this information that up to this path there is some node whose value is 30 or higher and my value is infinity or lower.

So far so good now, when I back up 25 what is going to change beta of this node is going to change. So this is the beautiful point where pruning happens. At this point, my alpha is 30 that is denoting that some node on this path has value 30 or higher. Its beta is 25. That means some node also has its value 25 or lower, this is incompatible. Alpha is greater than equal to beta. And as soon as this happens, I can prune the search and move up there is no need to discover, discuss anything anymore.

I have found that this is no longer the optimal branch. I should just go up. So, I will prune out any other children that 25 will have and send this information up. When I send this information of let us keep going so that you sort of complete this story when I send this information up. My max node is 30. But then when I send this information up to the min, its beta becomes 30. I sent this information down, so everybody's beta is now 30. As soon as I get 55, beta remains 30, then it gets 20, then beta becomes 20.

When I send this information up, alpha becomes 20, for the parent node, so let us say we are here, we have just seen 5, and we have just seen 5, so we send this information up. So my min node becomes 5 and what changes beta changes to beta become 5 now again, we are in this point, where alpha is greater than equal to beta. And because alpha is greater than equal to beta, we can prune this node.

So the 2 nodes that we intuitively figured out that we can prune we have been able to prune by this algorithm by checking if alpha is greater than equal to beta. I want to do one more. So we go up and we send this information up. So my alpha for the root node becomes 20. Basically it says, it knows that if I go left I can at least get 20. Now let us see right gives me a better value. So it is 20 or higher. I send this information down.

And I have just evaluated let us say this sub tree. In fact, I keep doing this I want to show you one other interesting example so now let us evaluated this sub tree. So my max node has value 15. It sends this information up. Its beta becomes 15. Now let us think about this and paused for a second. What has happened, the left sub tree of 20 has been fully explored. So it knows that it has a solution. The best solution is 20.

In other words if max decides to go left and min decides to go right and max decides to go left and when min decides to go right it can get to the leaf with value 20 so, the value of max is going to be 20 or higher. On the other hand, if we go to if max decides to go right then the min has explored the first sub tree and the value that it gets in the first sub trees 15. So now here is a point in the search space where I know that.

I have a solution 20 or higher but if I search in this sub tree of the right child of root, I will get a solution 15 or lower because it is a min node I already have got a solution of 15 there is no way I am going to get a solution of 15 or higher because beta is a min node so my solution is going to be 15 or lower. So, at this point my alpha is 20 and my beta is 15. And when this happens, the algorithm says, at this point in time alpha is greater than equal to beta.

Because alpha is greater than equal to beta, I have found a fallacy; there is no reason to search any further, because there is no way I am going to get a better value here. If this value is less than 15, this value min would be less than 15 max will still choose 20. If this value is greater than 15, the min will be 15 in which case as alpha will the max will still choose 20 there is no reason for the max to come to the right and therefore let us stop searching.

There is no point in searching further and let us back to track. And this algorithm is called alpha beta pruning in the context of mini max search, and this is where your last time. Now let us quickly try to analyze alpha beta pruning. So the first question I would ask is, would alpha beta pruning always helped? Or can it be sometimes not useful at all? Is it possible that it is never useful? To understand this question further, you have to realize that to learn about any node, I need to first explore its first child.

Once I have explored its first child, then I have got some temporary value. And once I have got this temporary value, I have got a better bounce. So I can say, next children I can prune. But until I explode the first child, I will not be able to prune anything out. I know am out of time. So I am going to stop here. I strongly encourage you to take a pause, go home and read on alpha beta pruning.

From the book or from any of your favorite resource, this particular method is a little involved. And it takes some time to get used to it. And if you are used to it if you understand this, then your next class when we discuss this further 2 days from now will go much better, I can assure you. So it is my request to you that before you come to the next class, please read up on alpha beta pruning and just do some practice so that you have some intuitions going. We will start from here, next class. Thank you.