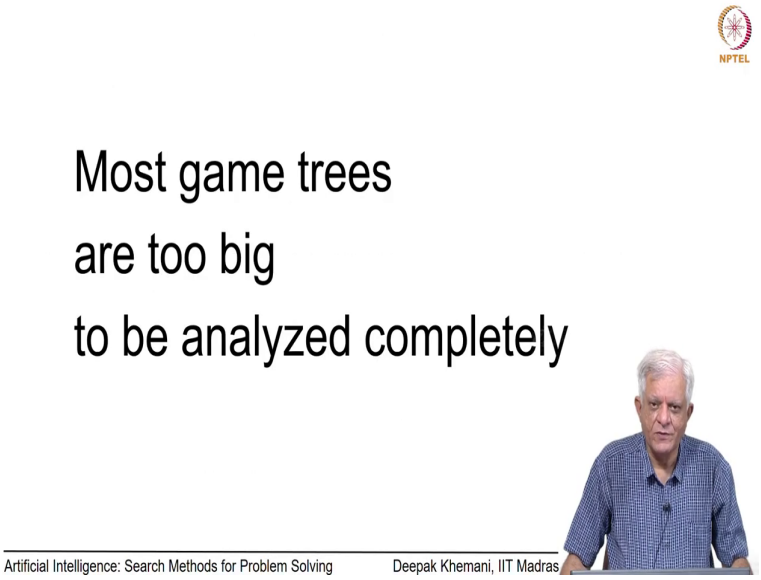
**Artificial Intelligence: Search Methods for Problem Solving**
**Prof. Deepak Khemani**
**Department of Computer Science and Engineering**
**Indian Institute of Technology, Madras**

**Chapter – 08**
**A First Course in Artificial Intelligence**
**Lecture – 53**
**Game Playing: The Evaluation Function in Board Games**

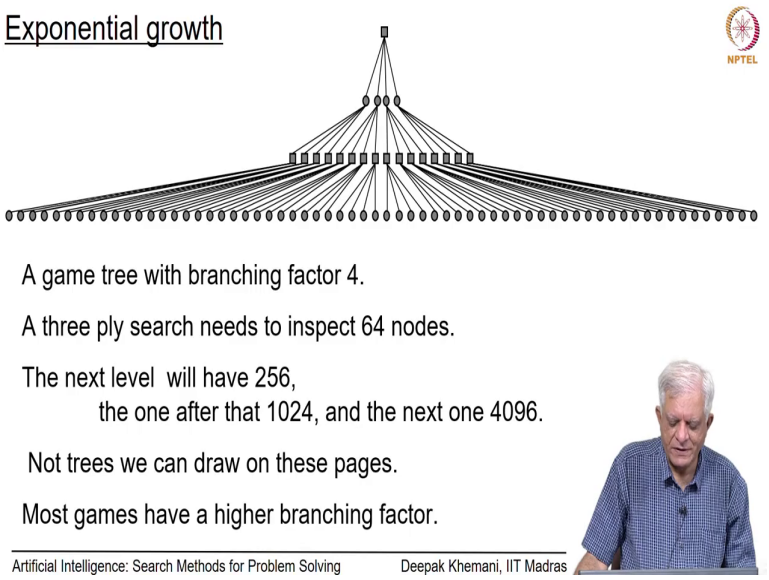
(Refer Slide Time: 00:14)




So, welcome back, we have been looking at Games; we started by looking at game theory and observed that these are models of rational behavior in multi agent environments. And then we narrowed down our focus to board games, which are two person give us some complete information alternate move games, deterministic games. And we looked at how there is a notion of a mini max value for such games and how to compute that value.

We also made an observation that, real games like chess and go are too large to be analyzed completely. As a consequence of which we do not know, what is the mini max value of a game like chess? But if you want to write programs to play those games; then we have to adopt some approaches, where you do not have to analyze the complete game tree and we do that by doing a primary look ahead.

(Refer Slide Time: 01:25)

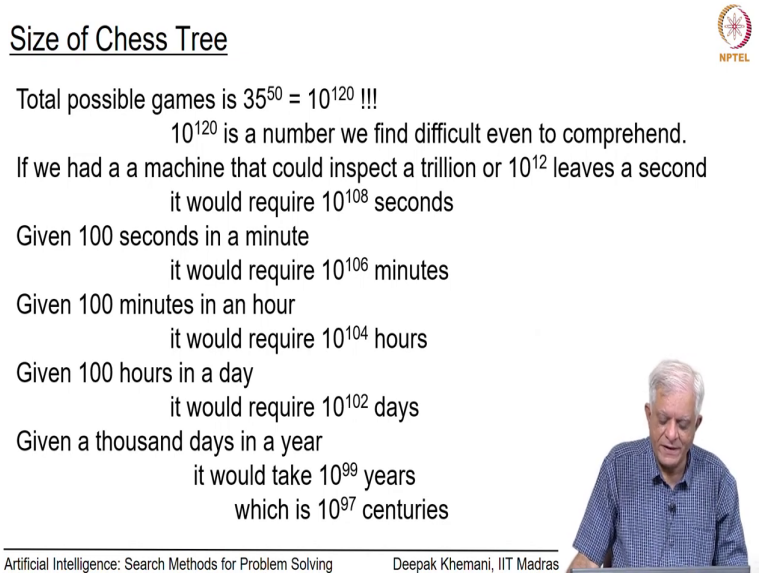



So, let us start on that process and see how we can tackle real games without having to wait for eternity till we have analyze the game. The main problem is that, there is exponential growth; that each player has a certain number of choices and at the next level, the next player has that many choices for each move made by the first player. And in that sense, as we have already seen in other situations also; the game tree grows exponentially.

So, supposing there was a simple game in which each player had four choices. So, the branching factor of the tree is 4; then a three ply look ahead search tree would inspect 64 nodes, then is the next level there would be 256, after that 1024 and 4096.

Now, these are trees which we obviously cannot draw on these pages and also we cannot analyze them if they are long games; long games meaning the number of moves that a game has. And most games as we have observed have higher branching factors.

(Refer Slide Time: 02:34)




So, let us go back to what we said about chess; we said that the total possible games is about 35 raise to 50, which translates to 10 raise to 120. And we said that 10 raise to 120 is the number that we cannot even comprehend; our human cognition is attuned to the world around us, and numbers like 10 raise to 120 do not make too much sense.

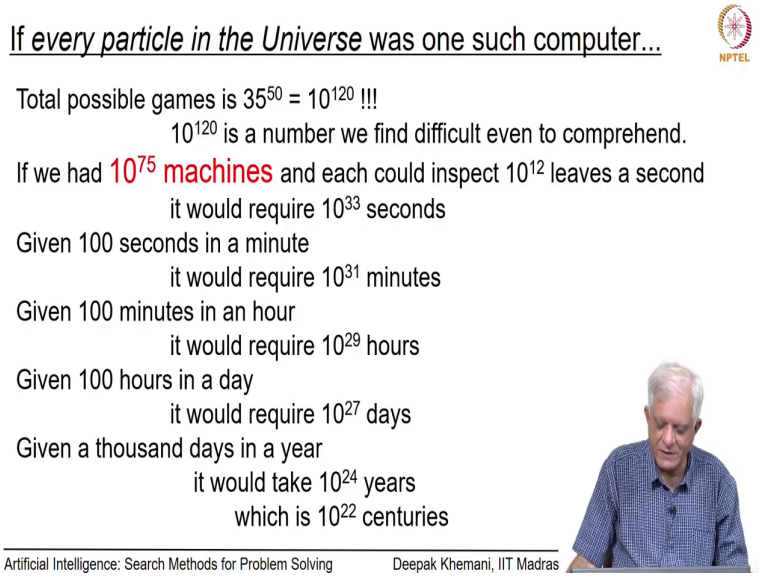
So, let us assume that we had a very powerful machine that could inspect a trillion leaves a second, that is 10 raise to 12 leaves a second. And as you can see that, such a machine would required 10 raised to 108 seconds. So, all you are done is from 120, you have subtracted 12 and you have got 10 raise to 108 and it would require 10 raise to 108 seconds.

So, since we have working in powers of 10, we will you know assume that our calculations are all powers of 10. So, we will assume that a second has 100, a minute has 100 seconds and hour has 100 minutes and so on; just to take a more conservative approach and see how long it would take for us to compute or analyze the complete chess tree.

So, if we had 100 seconds in a minute; then this time would reduce to 106 minutes and if you had 100 minutes to an hour, this time would reduce to hundred and 10 raise to 104 hours. If you had 100 hours in a day, this would become 10 raise to 102 hours. And if you had a thousand days in a year, this would become 10 raise to 99 years, which is 10 raise to 97 centuries. So, obviously, this is a numbers still beyond our comprehension.

So, let us assume that, compare it to the size of the universe for example, and it is estimated that the universe has about 10 raise to 75 or 10 raise to 80 fundamental particles. And if each of them were to be super computer which could inspect a trillion leaves in a second; then what would happen?

(Refer Slide Time: 04:56)



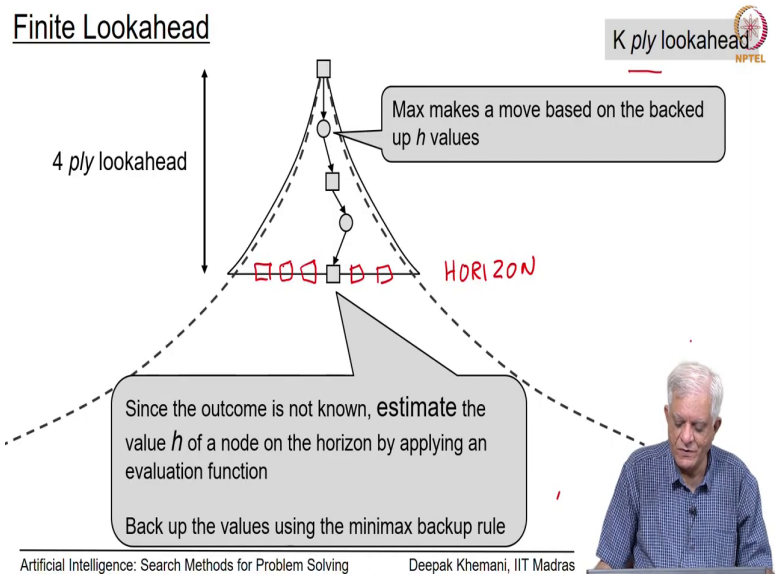

So, what we are saying here is that, if every particular in the universe was such a computer. Then the total number of games you have already said is 10 raise to 120. And if you had 10 raise to 75 machines and each could inspect 10 raise to 12 leaves a second, it would still require 10 raised to 33 seconds.

And then we go through our similar calculation as we did recently; if we had 100 seconds in a minute, it would need 10 raise to 31 minutes; if we had 100 minutes in an hour, it would need 10 raise to 29 hours. If we had 100 hours in a day, it would need 10 raise to 27 days.

And if we had a thousand days in a year, it would be 10 raise to 25 years, 10 raise to 24 years and which is equivalent to 10 raise to 22 centuries.So, again it is 10 raise to 22, that is one

followed by twenty two zeros and that is not the kind of time in terms of centuries that we are willing to wait.

(Refer Slide Time: 06:01)




So, what do we do? All game playing programs do is, they do a limited look ahead. So, if this is a game tree which is growing exponentially; we do a look ahead which is called a K ply look ahead.

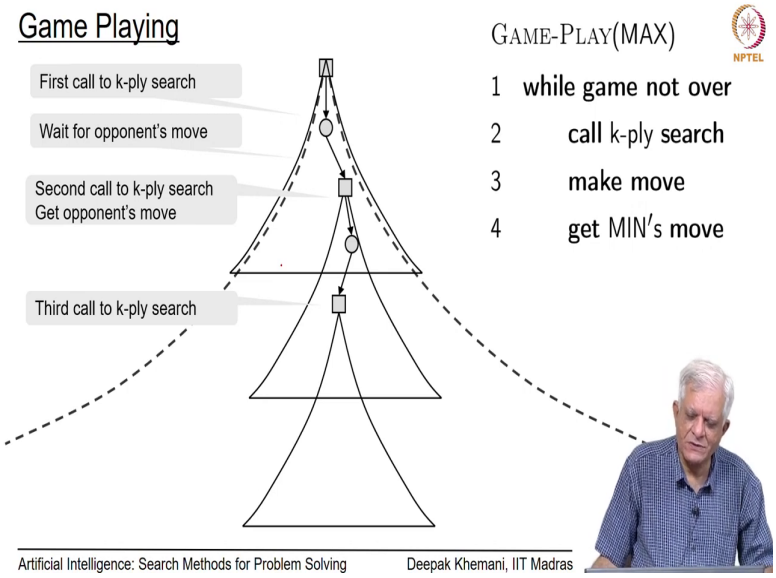
So, a ply is a name that we use commonly in game playing parlance; we say how many moves are we playing. So, K ply look ahead means that, you are looking at K moves; you make a move, your opponent makes a move, you make a move and so on and you are only looking up, up to K moves ahead.

So, for example, if you are doing a 4 ply look ahead; then we are looking at part of the game tree and not looking at the complete game tree. So, what do we do at the end of these 4 ply's, which is called the horizon? We have reached some board position, but the board position is not a completed game. So, we do not have the outcome of the game; we do not know whether max is winning or max is drawing or max is losing.

So, instead we do what we often do in air, which is to apply some sort of a estimating function or a heuristic function. And for any board position on the horizon, we estimate the value of, value h and we will use the term h for this value carried forward from the idea for heuristic function in search. And we estimate this value h by applying a evaluation function.

And after we have applied the evaluation function, we back up. So, we have apply the evaluation function to all the nodes in the on the horizon. So, I have not drawn all of them; but you can imagine that at all the nodes, you apply the evaluation function and then you back them up. And based on the backed up value, max decides what is the move to be made essentially.

(Refer Slide Time: 08:04)




So, how does game playing work? The basic idea of game playing is not to analyze the whole tree in one shot, but to do this every time you want to make a move. So, the high level algorithm for playing a game for max and we have said that we will always look at things from the perspective of max; is that while the game is not over, you call your k ply search algorithm which will look ahead up to k ply's, apply the evaluation function, backup the values and choose the best move, based on that you make your move.

And then you wait for MIN's move; MIN is your opponent. And when min makes a move, then you again call for k ply search. So, the game playing proceeds as follows. So, this larger envelope that we see is the game tree; but we are not going to explore the entire game tree by any means, we are going to explore small portions of it up to a finite look ahead. And first we

as we have just observed call k ply search and based on that k ply search, we make one move, ok.

After having made this move, we wait for your opponent to make the move. And very often people try to figure out what to do in this time when you are waiting for opponents move; but we, we will see, we may get back to this a little bit later. But so, this is the first cycle that of this game playing which is over, which is that you have made a move after doing k ply search and the opponent has made a move by whatever means the opponent is adopting. And now, again you have to make a move.

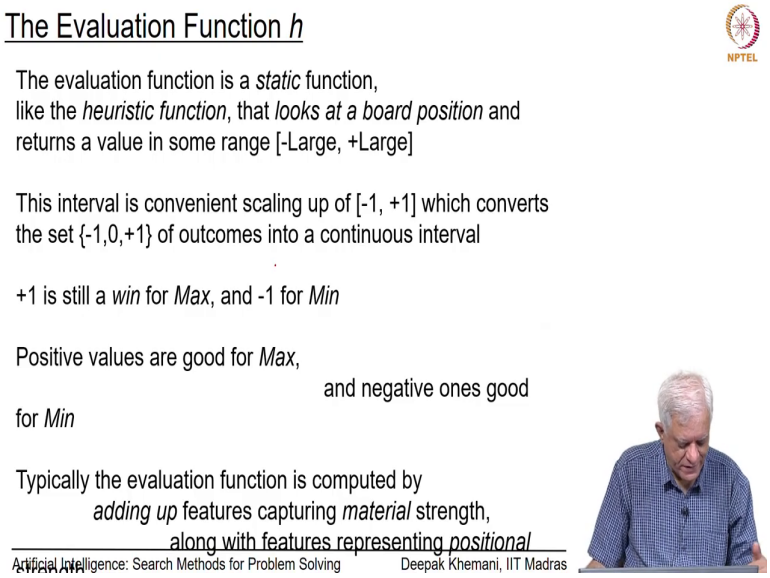
Now; obviously, this is a node that you have seen earlier in your k ply search; but now you want to again look at k ply's ahead from this particular position. So, what do you do? You do a secondary, second search; you again do k ply search from here, you again make your move, you again wait for your opponent moves and then when you have to move again, then you again call k ply search.

So, in this manner for every move that you make, you will call k ply search and in some sense, you will look that much ahead, so two moves ahead in every search. So, in the first move you, in the first search you made a move, opponent made a move; then when you do again k plys, that means you are looking two more moves ahead and in that sense you keep looking further and further into the game.

And hopefully far enough so that, you get a good sense of where the game is going essentially. What should k be? Now, k typically depends on the computing resources that you have and also the time that you have for playing the game essentially.

Now, if you are writing a chess playing program that was playing in a tournament; then you are allotted the limited number of time to make a certain number of moves, and based on that you would make your, you would decide how much should the k ply be essentially.

(Refer Slide Time: 11:20)




So, what is a, what is this evaluation function that we are talking about? How do we, what does it do for it? It is exactly like a heuristic function that we saw in heuristic search; that it is a static function and by a static function we mean that, it only looks at a board position and does not do any look ahead or search from there, like the heuristic function that we had in our heuristic search algorithm.

So, it looks at a board position and returns a value in some range, which we will call as minus large to plus large. So, in, so this minus large is an approximation we use when we are writing algorithm to say that, it can be as small as possible and plus large means that as large as possible essentially.

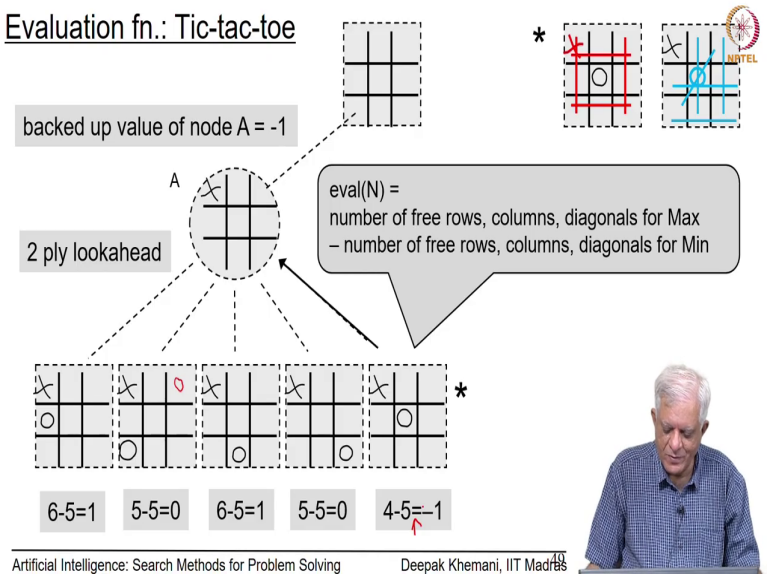
So, that will obviously, depend upon your design of your program. This minus large to plus large interval is just a convenience scaling up of the interval minus 1 to plus 1 the real

interval, which we are interested in, which converts the set of three choice three outcomes in the game which is minus 1, 0 and plus 1. Remember minus 1 is a loss for max, 0 is a draw for both players and plus 1 is a win for max.

So, these are the three possible outcomes. But now we are having our evaluation function; return a value in the range minus 1 to plus 1. Plus 1 is still a win for Max and minus 1 for Min as before. So, these are the extremes of our interval; but in between any values, positive values are good for Max and negative values are good for Min.

And typically the evaluation function is computed by adding a features capturing material strength, along with features which represent positional strength. So, this text has gone a little bit out of the slide; but the last line should reach along features representing positional values essentially.

(Refer Slide Time: 13:26)



So, let us look at some of them. Let us consider a simple game the Tic tac toe game which children play, also known as cross and naughts, where alternately you put a cross or a naught; naught meaning a zero on one of the nine squares that are available to you.

So, this is a starting position in which the first player Max has to move and Max has to decide where to put a cross. So, that is a task that Max is addressing. So, let us assume that Max is doing 2 ply look ahead; which means that max is looking ahead two moves. And so, first Max considers whether to put a cross at the corner of the game, corner of the board.

And that is the first move that Max is considering. And then for this move that Max is considering; it considers all possible moves that Min can make. So, Min is putting a naught or a circle on the board and we have just drawn all the distinct positions to at which Min can play the naught. In practice of course, there are eight possible positions in which Min can play a naught; but some of them are symmetric with respect to each other.

So, for example, in the second board position; if Min were to play a naught here, then the two positions are strategically identical. So, we kind of somehow ignore symmetries in our analysis; though when we are writing a program, we may or may not ignore symmetries.

So, in any case, there are these five distinct moves that Min can respond to by in response to Max's first move which is let us call it the node A essentially. So, in response to A, Min can make any of these five choices. And since Max is doing a 2 ply look ahead, at this stage max applies evaluation function.

And let the evaluation function be as follows that, the evaluation function counts the number of free rows or columns or diagonals which are available for Max. And from that you subtract the number of free rows, columns and diagonals available for Min; remember that the game of cross and naughts is one when someone gets three in a row or three in a column or three in a diagonal.

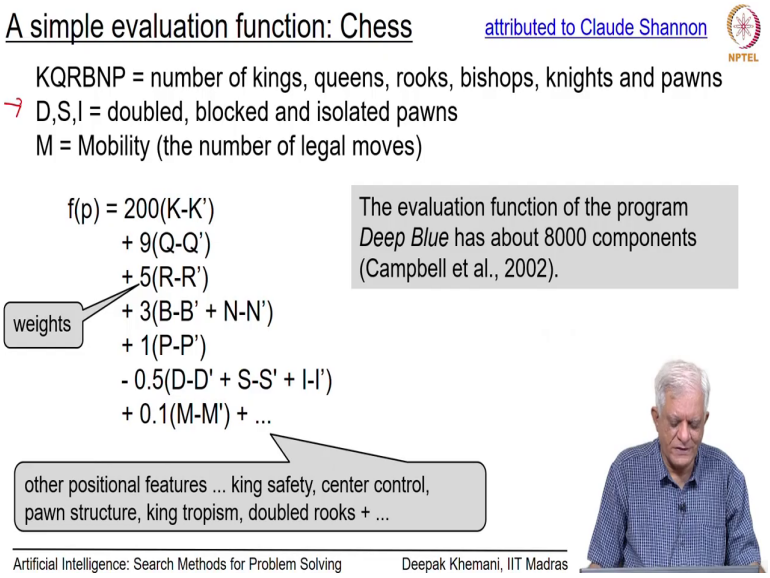
So, we are just trying to see, how much is the possibility for each of these two players. So, let us look at value for the last board position; in the last board position, we have Max has played at the corner and Min has played at the center.

So, if you look at this board position based on the evaluation function that we have; then you can see that these are the following possibilities for the two players. For Max we have shown in red that, there four possible rows or columns or diagonals available, where Max could possibly win a game. But for Min we have five of them. So, those five green lines that you see on the right most in this thing, they represent the five possibilities for Min.

So, the evaluation function for Max for this node evaluates to 4 minus 5 which is minus 1 essentially. So, in this manner we keep applying the evaluation function to each of those five board positions on the horizon, and then we know that we are doing a mini max backup rule. So, the best value or the best value from Min's perspective which is the lowest value amongst these five values.

These values are 1, 0, 1, 0, minus 1. So, since A is a Min node, Min would choose the smallest of this and therefore, this value would be backed up for the, for the node A and we would back up the value of node A as minus 1, ok. So, this illustrates the evaluation function for the simple game of Tic-tac-toe. Let us now look at perhaps the most possible game that has been used in for programming, which is chess.

(Refer Slide Time: 17:47)




And this depicts a simple evaluation function for chess; this evaluation function has been attributed to Claude Shannon, who was one of the pioneers of computing known as the father of information theory.

And this evaluation function which he called as f of p or function of a position p is made up of a set of components essentially. And the components are as follows that, the letters K Q R B N, P they counts the number of kings, queens, rooks, bishops, knights, and pawns that each player has; K is for Max, and K prime is for Min essentially.

So, obviously, in the case of king and queen, at least in the beginning of the game we have only one queen; towards the end of the game you may have two queens if your pawn gets

promoted to a queen. But in the beginning, you may have one extra queen; if you have if both of you have a queen on the table, then you know it amounts to equal.

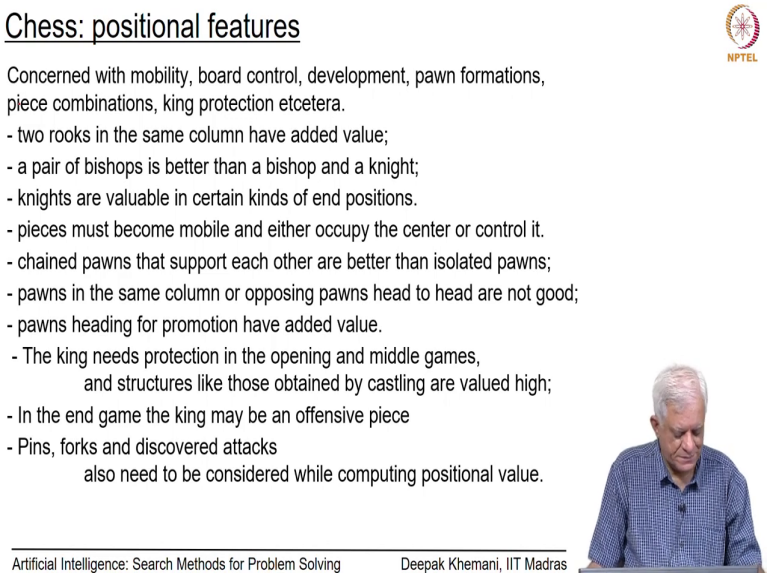
So, Q minus Q prime would be zero; but if one of you did not have a queen, then you know it would add up to the evaluation function. The numbers that you see that that you are multiplying these differences with 200, 9, 5, 3, 1 and so on; these are the weights that you assigned to each pieces essentially.

So, many chess players would say that a rook has a weight of 5; on some scale on which queen has a weight of 9, the bishop and the knights have weights of 3, pawn has a weight of 1 and so on. So, apart from material value, we also have positional value as we have said and the positional value here is indicated by these three features D, S and I which talks about how the pawns are organized; double pawns or blocked pawns or isolated pawns.

And M is the mobility, the number of legal moves that are available to each. So, the more moves you have, the more the choices that you can make and the less constrained that you are. And these mobility features which have not been listed here completely, have things like king safety, central control of center, pawn structure, king tropism, double rooks and so on.

And you can devise more and more such patterns or features to evaluate the board position of a game. And the game, the program Deep Blue which was a program which beat Kasparov in 1997, has about 8000 such components, 8000 different such components. And this is according to the paper that was published by the author of the program, one of the authors of the program Campbell in 2002.

(Refer Slide Time: 20:50)




Let us look at what else you can do in chess. So, we said that positional features are important. So, essentially, how do strong chess players look at the board position and say, oh this is good for Max or good for white or this is not so good for white and things like that? The positional features are concerned with mobility, board control, development, pawn formations, piece combinations, king protection etcetera.

So, some other features that chess players or chess programmers have used are as follows; for example, that two rooks in the same column have added value, because you know the rooks can move either vertically or horizontally and if they are in the same column, that means they are protecting each other.

So, the other rook can kind of freely move up and down. A pair of bishops is better than a bishop of a knight. So, those of you who have tried to do end games would observe that this

is the case; that with two bishops you can check mate and opponent, but with the bishop and knight you cannot.

Knights are valuable in certain kinds of end positions and pieces must become mobile and either occupy the center or control it. So, center control has been a very focused point for chess player. So, chess enthusiasts very often, they start off by looking up a book like modern chess openings, which describes the result of analysis by hundreds of strong players, as to what are the ways to start opening up the game.

So, we had observed that Max has 20 moves to start with in chess; but very very rarely would more than 4 or 5 of these 20 moves be used by any good player. That is because the cumulative experience of chess players has resulted in, in the knowledge that certain moves are good, and the focus of such things has often been to control the center.

So, because center is where you know the maximum action can take place. So, if you want to control the center, you your pieces must be either occupying the center or they must be controlling the center remotely. So, for example, a bishop can be attacking some centers squares or a knight or a queen and so on.

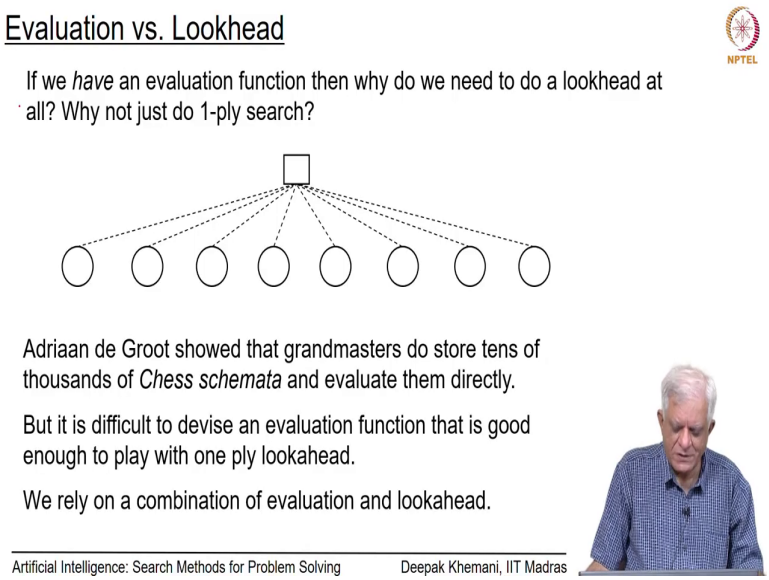
Control or center has been very strong focus point essentially. Then chained pawns that support each other are better than isolated pawns; because you know you know the opponent cannot just freely grab a piece that you have on the board. Pawns in the same column or opposing pawns head to head are not good, because their mobility is reduced.

Pawns heading for promotion have added value. So, if you remember the rules of the game chess, which we have not discussed; but I hope you know them that if a pawn reaches the last square, the last row in a chess, then it can get promoted to any piece that the player chooses. Most often it is a queen; but in some peculiar cases, it may not be the queen and something else.

The king needs protection in opening and middle games. So, a focus of chess openings is to first somehow secure the position of your king and there are various kinds of strategies that people use like castling for example, and these give high value to both positions.

But in the end game, when the number of pieces are small; the king may be an offensive piece and it has a different values essentially. So, a constraint king may not be good, but a freely moving king maybe good essentially. Things like pins, forks, discovered attacks also considered while computing positional value essentially.

(Refer Slide Time: 24:56)




We will look at forks in a moment. One question that you might ask is, if you have any valuation function; why do you need to do look ahead at all? Why not just do one ply search

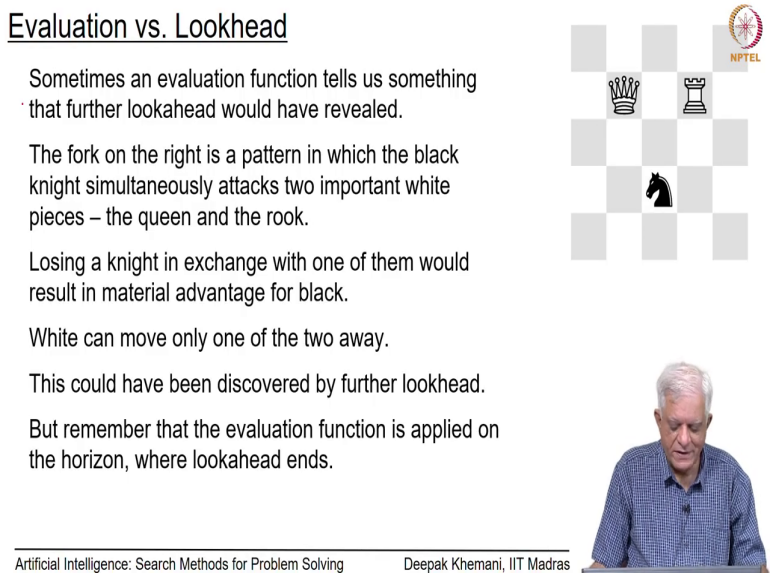
as this diagram shows? That Max considers all the possible moves that Max can makes; apply the evaluation functions each of these moves and chooses the best one.

Now, this would require that your evaluation function is very good essentially. Now, the chess psychologist de Groot had done a study in which he had observed that, grandmasters store tens and thousands of chess what he called as schemata. So, these are patterns which they have kind of learned through experience, and simply by looking at a chess position; a grandmaster can say this is good, this is not good and so on.

And the and the way they do it is, because they have accumulated their experience of chess or knowledge of chess into schemas or schemata's, which they can spot immediately essentially.

So, there is always a tradeoff between how much you can recognize by looking at a board position versus how much do you need to kind of look ahead and try to figure out what is going to happen essentially. The trouble is that, it is difficult to devise an evaluation function that is good enough to play with one ply look ahead essentially. We rely on the combination of evaluation and look ahead therefore, essentially.

(Refer Slide Time: 26:29)




So, let us look at some situations, where what we are doing by evaluation could have been got by look ahead. Here is a fork, as this as it is called in the game of chess. So, it is a fork, because the knight, the black knight can attack is attacking both the white queen and the white rook.

And if you have a pattern which says that you give certain value to fork, if you can see a fork like this; then it actually gives us some information which further search would have revealed essentially.

So, in this pattern as we said; the black knight is simultaneously attacking two white pieces, the queen and the fork. Now, if you remember we briefly mentioned what the valuations of

these pieces have; the queen is was 9 points, the rook was 5 points, and the knight was only 3 points.

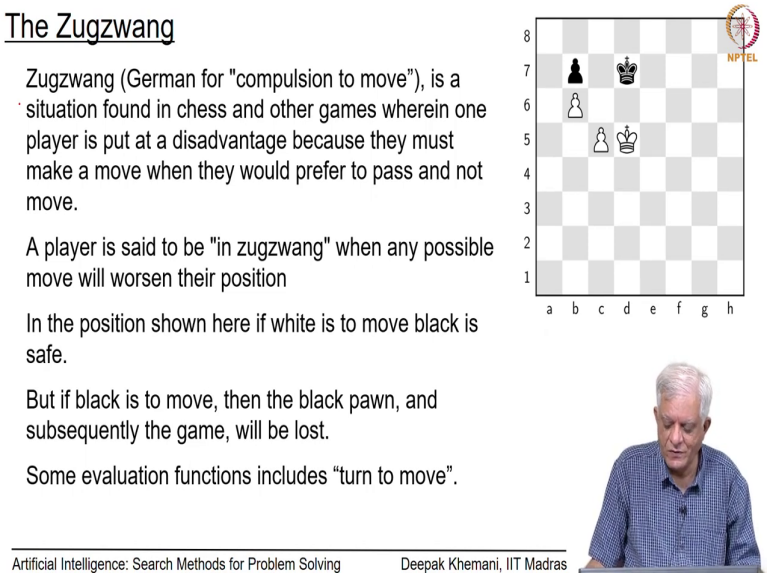
So, if the knight could if the, if the black player could exchange the knight with the rook; then black player would happily do that, because it would give the black player material advantage.

The white player can only move away one of the two pieces. So, that is why it is called a fork that, white has limited choice which is to either move away the queen or move away the rook essentially. Now, this could have been discovered by further look ahead.

If you had you know done a little bit of look ahead that white plays the queen somewhere and then the black knight captures a rook and somebody captures the black knight and then therefore, this is the final material position. But remember that the evaluation function is applied on the horizon; horizon is a place where look ahead has stopped.

So, if you can catch that pattern at the horizon itself; that means you have got some something extra from looking at the board position, which you would have otherwise got if you had the freedom to look ahead further essentially.

(Refer Slide Time: 28:37)




Some very popular chess positions are called zugzwang. So, the zugzwang is a German word; it is using chess and in other games also, where one player is put at a disadvantage, because they must make a move when they would prefer to pass or not move.

So, we will not discuss this in too much detail here; but if you look at this chess book position shown here, then you can see that white has got two pawns and the king, black has got one pawn and a king

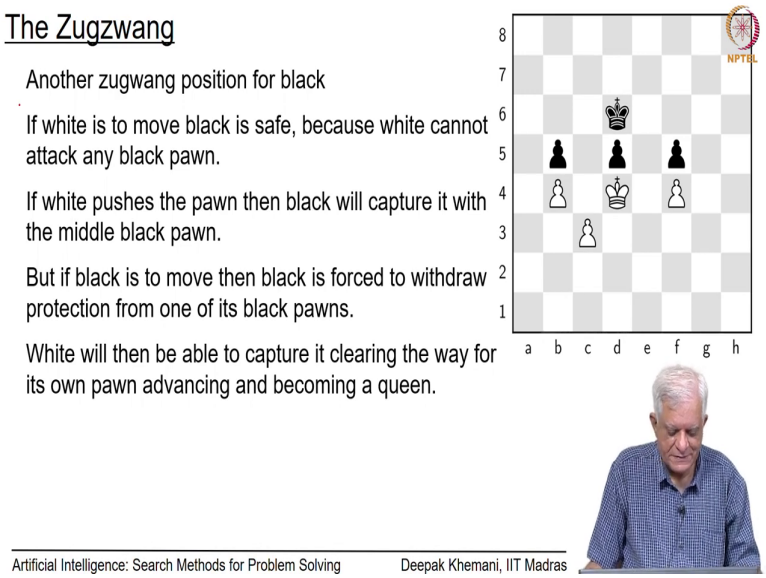
If somehow white could move forward and promote one of his pawns to a queen or something; then white would eventually win the game. But the situation is such that, black is preventing white from coming forward essentially. So, a player is said to be in zugzwang when any possible move will worsen the position.

Now, in this both position what this really critical is, whether it is white to move or whether it is black to move essentially. Now, if white is to move, then black is safe; because the only thing that white can do is to move the king somewhere and black can also do some shadow moves and keep coming back to the same position.

White cannot move the pawn, because a pawn would be captured by the black pawn and then if white moves the other pawn ahead, the black king can go after it and capture it essentially.

So, if white is to move, then black is safe; but if black is to move, then the only move that black can make is to move the king. Then if black moves the king, then white will be able to capture the black pawn, and eventually win the game essentially. So, sometimes in the evaluation function, we also include the fact as to whose turn it is to move essentially.

(Refer Slide Time: 30:36)



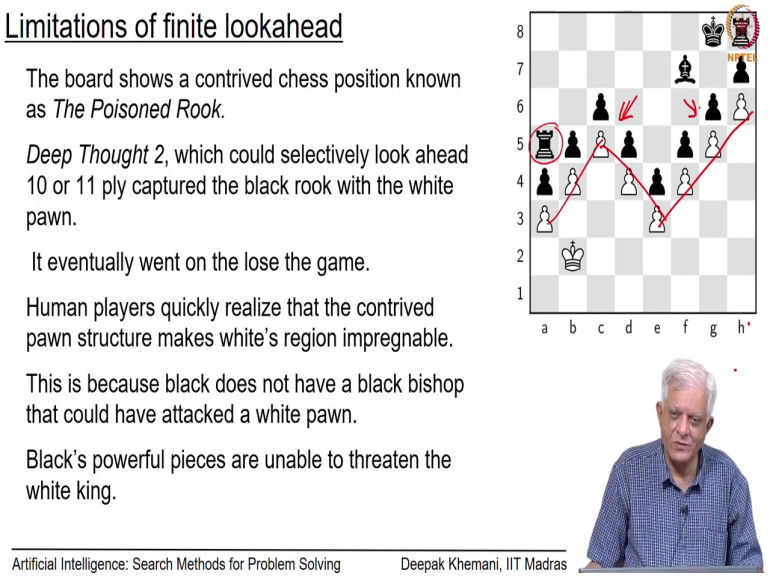
And many chess programs do that. Let us look at another slightly more sophisticated zugzwang position. The position is similar in the sense that, the black king is trying to protect three pawns and the white king is trying to attack one of those pawns, so that white can go ahead and promote a pawn.

The material strength is equal, but the positional strength is different as you can see. Now, in this position black is again at a disadvantage, if black were to move; if white is to move, black is safe, because white cannot attack any black pawn essentially.

White, so the again the king would have to retreat and black could measure the move or something like that. But if black, if white were to push upon; it would get captured and that would be the end of the story for white essentially.

But if black is to move; then black is forced to withdraw protection from one of the three pawns. And then again white would go ahead and capture the that pawn clearing the way for some of his pawns to advancing in becoming a queen. So, again this kind of a position is critical to as to who has to move essentially.

(Refer Slide Time: 31:47)




Let us look at one more position, and here we want to illustrate that; because of the fact that you are doing a finite look ahead, you can sometimes not come to the right conclusion, and this is simply because you are doing a finite look ahead.

Now, this is a very contrived chess position; it was created specifically to test computer programs, and this position is known as the poisoned rook. So, if you look at this position carefully, you will see that there is a kind of a fortress in which white is living of black pawns and none of the a white pawns. And none of the black pieces can you know penetrated this safe area that white has.

But now black has very cleverly offered a rook as sacrifice to white. Now, if you were to simply go by material advantage which is what chess programs do; then capturing a black rook would definitely be worth its effort. And that is what this program called Deep Thoughts

2, which could actually selectively look ahead up to 10 or 11 ply capture the black rook and with the white pawn.

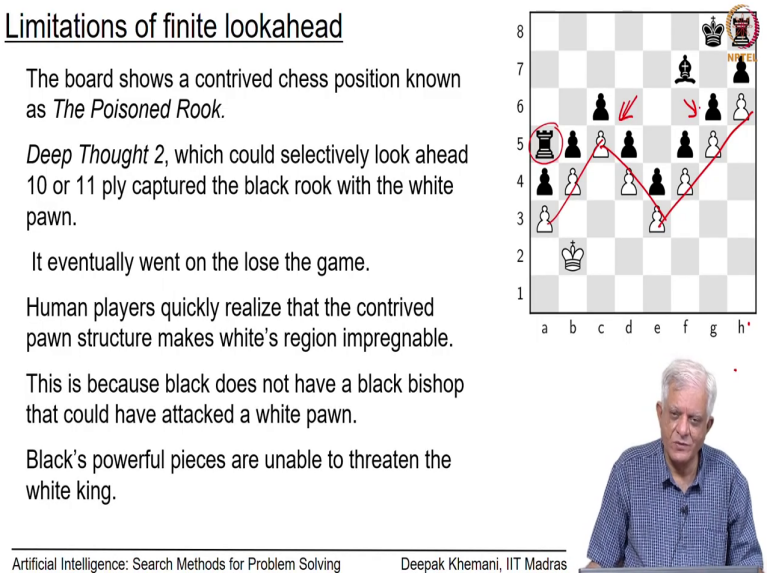
Once that white pawn which captures the black rook is removed, it paves the way for other white pieces to come forward and eventually defeat other black pieces to come forward and defeat in this thing. So, eventually deep fritz also went on to lose the game.

But human players quickly realize that, this very contrived structure you will rarely have eight pawns on each side in such end positions. But in this structure any human would recognize that, all white has to do is to simply move around the king in its region, and there is nothing black can do to win the game.

If black had a black bishop; then the black bishop could have possibly attacked one of the white pawns, either this one or this one or any of the white one. Because they are all sitting on black squares, but black does not have a white, black bishop in this case.

So, this illustrates that a finite amount of look ahead, in which you are looking at material value and positional value is often unable to discern some odd situations like this which human beings can still do quite well, ok.

(Refer Slide Time: 34:33)




So, in the next class, we will start looking at how to analyze game trees with the finite look ahead. So, here we have a small game tree, where each player has two choices at every stage and we are doing a four ply look ahead, and at the bottom are the leaves to which the evaluation function has been applied. So, we will take this up in the next session.