Unit -2 Game Playing

Why to study Game Playing?

1) Its a good reasoning problem, formal and nontrivial
2) Direct comparision with humans and other computer programs is easy

→ Mainly games of strategy with the following characteristics

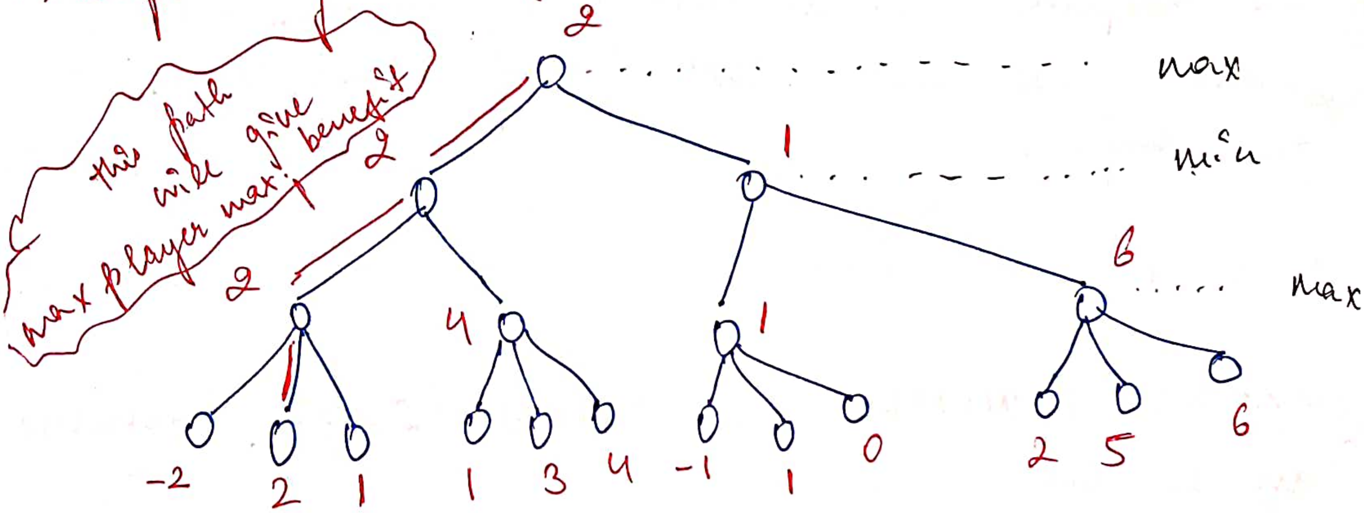
1) Sequence of moves to play.
2) Rules that specify possible moves.
3) Rules that specify a payment for each move
) Objective is to maximize your payment.

Min - Max Algorithm

- Min-max algorithm is recursive or Backtracking algorithm which is used in Decision making and Game Theory. It provides an Optimal move for the player assuming that Opponent is also playing Optimally.
- This Algorithm uses Recursion to search through the Game tree
- This Algorithm is mostly used ~~in~~ for game playing in AI such as Chess, Checkers, tic-tac-toe, go and various there games.
- The Algorithm computes the minimax decision for the current state.
- In this Algorithm 2 Players play the game, one is called MAX and other is called MIN.
- Both the players fight it as the Opponent player gets the minimum benefit while they get the Maximum Benefit.
- Both players of the game are Opponent of each other, while MAX will select the maximized value and min will select the minimized value

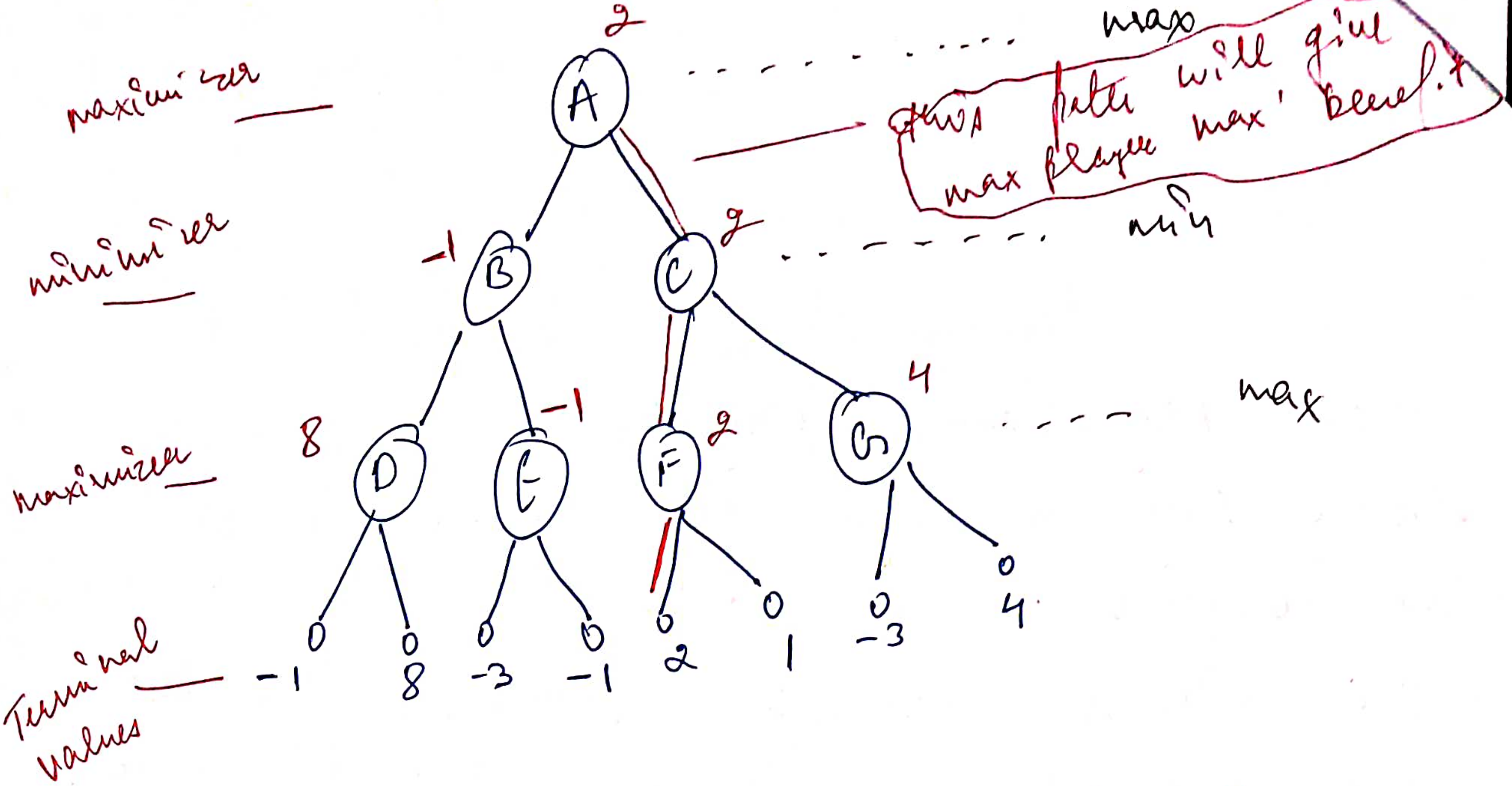
- This Algorithm perform a depth-first Search Algorithm for the Exploration of the complete Game tree.
- The minmax Algorithm proceeds all the way down to the terminal node of the tree, then Backtrack the tree as Recursion.
- Initial value of maximum $= -\infty$ } worst
  Initial value of minimum $= \infty$ } values

Example of MiniMax Theorem



Initial values [max $= -\infty$
min $= \infty$

- At 1st stage we need to find max so of it will compare $(-2, -\infty) \to (-2, 2)$
$\to (2, 1) - 2$



Q why do we take the min value every other level of the tree?

- These ~~moves~~ nodes represents the opponents choice of move.
- The computer assumes that the humans will choose that move that is of least value to the computer.

Properties of Minimax

1) Complete : It is complete, it will definately find a soln (if exist), in the finite search tree.

2) Optimal : Optimal if both opponents play optimally.

3) Time complexity : As it performs DFS for tree, so $O(b^m)$ – b → branching factor, m → maximum depth.

4) Space complexity : similar to dfs $O(bm)$

• Chess

b = 35

m ≈ 100 (game length)

$b^m = 35^{100} = 10^{154}$

infeasible

• For all practical purposes, Minimax is not the helpful algorithm because we will not be able to play it till the very end, so we need to come up with the traits of trick so that we can search till limited depth and still figure out the good soln

We will see 2 ideas

1) Not searching things which are provably sub optimal

2) Cutting off the search and do machine learning to help with utility f^n estimation.

Unit - 2 Topic Alpha Beta Pruning

- Alpha - beta pruning is a modified version of the minimax Algorithm. It is an Optimization technique for the minimax Algorithm.
- As we have seen in the minimax search Algorithm that the no. of game states it has to examine are exponential in depth of the tree. Since we cannot eliminate the exponent, but we can cut it to half.

→ Hence there is a technique by which without checking each node of the game tree we can compute the correct minimax decision, and this technique is called pruning.

- It involves 2 threshold parameters as:

1) Alpha : The best (Higest value) choice we have found so far at any point along the path of maximizer. The initial value of Alpha is $-\infty$.

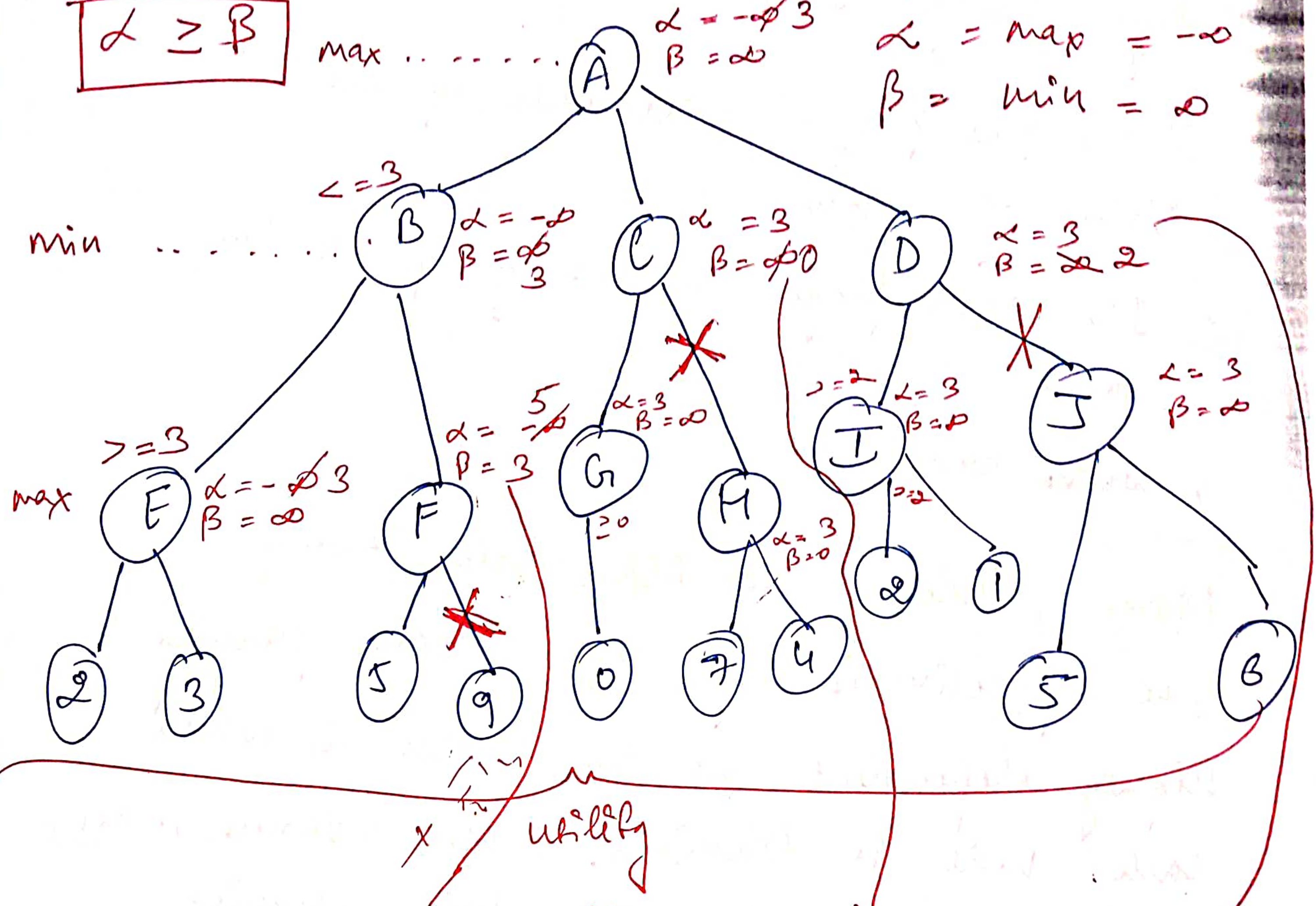
2) Beta : The best (lowest - Value) choice we found so far at any point along the path of ~~Maximizer~~ minimizer . The initial value of beta is $+\infty$.

- Alpha - beta Pruning can be applied at any depth of a tree , and sometimes it not only prune the tree leaves but also entire sub - tree .

- The Alpha - Beta pruning to a standard minimax Algorithm returns the same move as the standard Algorithm does , but it removes all the nodes which are not really affecting the final decision but making Algorithm slow . Hence by pruning these nodes , it makes the Algorithm fast .

DETAILED LECTURE NOTES

$\boxed{\alpha \geq \beta}$



Here we will compare here for the α & β as $\alpha \geq \beta$

$5 > 3$ ✓

$\alpha \geq \beta$
$3 \geq 0$

$\alpha \geq \beta$
$3 \geq 2$

No. of α cutoffs = 1

~~α = 1 cutoffs~~

No. of β cutoffs = 2.

Condition for Alpha - Beta pruning

$\alpha >= \beta$

Key points about Alpha - beta Pruning:

- The Max player will only update the value of Alpha.
- The Min player will only update the value of beta.
- ~~which Write~~

Move Ordering in Alpha Beta pruning

The effectiveness of Alpha - beta pruning is highly dependent on the order in which each node is examined. Move order is an import aspect of Alpha beta pruning.

It can be of 2 types

1) Worst Ordering: In some cases, this algorithm does not prune any of the leaves of tree, and works exactly as minimax Algorithm. In this case, it also consumes more time because of Alpha - beta factors such as move of pruning is called worst ordering.

In this case the Best move occurs on the right side of the tree. The Time complexity for such an order is $O(b^m)$.

2) Ideal Ordering : This occurs when lots of pruning happens in the tree, and the Best moves occur at the left side of the tree.
We apply DFS hence it first search left of the tree and go deep twice as minimax Algorithm in the same amount of time. Complexity here would be $O(b^{m/2})$.

Rules to find good Ordering :
1) Occur the Best move from shallowest node.
2) Order the Nodes in tree such that the best nodes are checked first.
3) Use Domain Knowledge while finding the Best move. Eg: for chess, try order : captures 1^{st}, then threats, then forward moves, backward moves.

4) We can bookkeep the states, as there is possibility that states may repeat.