MINMAX

- Min Max Algorithm is a recursive or backtracking algorithm which is used in decision making and game theory.
- In game playing it provides optimal move for the player assuming that opponent is also playing optimally.
- Min-max algorithm uses recursion to search through the game tree.
- Min-max algorithm is mostly used for game playing in AI. Such as Chess, Checkers, tic-tac-toe, and various two players game.
- This algorithm computes the minimax decision for the current state.
- In this algorithm two players play the game, one is called MAX and the other is called MIN
- Both the player 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, where MAX will select the maximized value and MIN will select the minimized value.
- This algorithm performs a depth first search algorithm for the exploration of the complete game tree.
- This algorithm proceeds all the way down to the terminal node of the tree, then backtrack the tree as the recursion.

Terminologies

Game tree : In form of tree structure consists of all possible moves (move from one state to next state)

Game can be defined as a search problem with following components.

Initial state : Starting board position

Successor function : Defines what a legal move a player can make.

Terminal state : Position of the board when the game gets over.

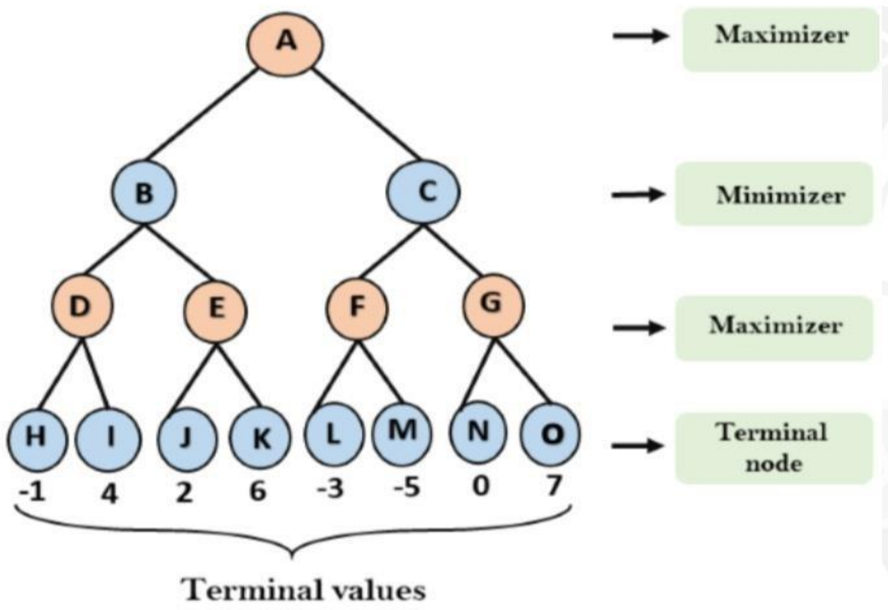
Utility function : Assigns the numeric value for outcome of a game.

Working of Algorithm

- For chess or tic-tac-toe outcome can be win, loss, draw.
- Max player tries to get highest possible score, Min player tries to get lowest possible score. Both player act opposite to each other.
- At the terminal node , the terminal values are given so we will compare those value and backtrack the tree until the initial state occurs.
- Following are the main steps in solving the two player game tree:

Step 1:

- In the first step the algorithm generates the entire game tree and apply the utility function to get utility values for the terminal states.
- In the below diagram, lets take A is the initial state of the tree. Suppose maximize (Player1)takes first turn, which has worst case initial value -∞ and the minimizer(Player 2)will take next turn which has worst case initial value +∞.



Step 2:

Now, we find the utilities value for the maximiser, its initial value is -∞. so we will compare each value in the terminal state with initial value of maximize and determines the higher nodes values. It will find the maximum among the all.

For node D

max(-1, -∞) =>max (-1,4) =4

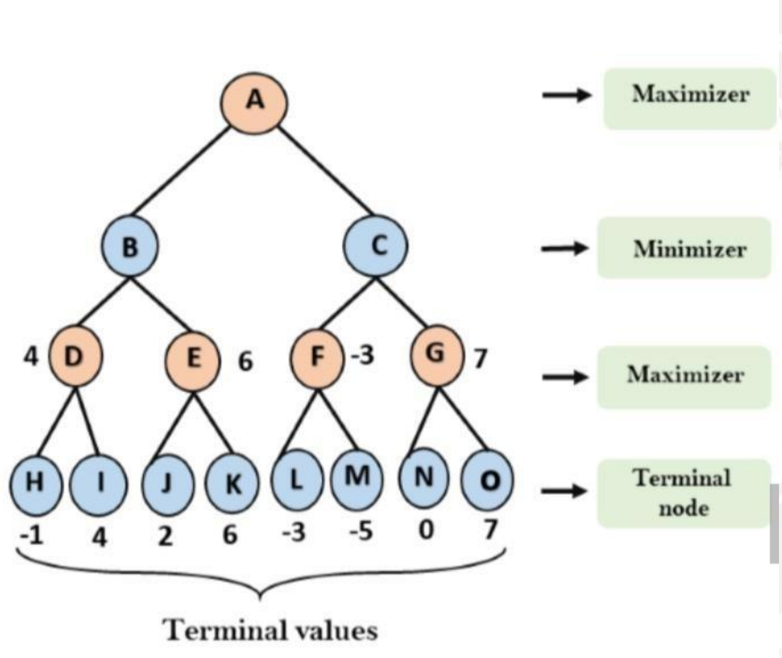
For node E

max(2, -∞)=> max(2,6) =6

For node F

max(-3, -∞)=> max(-3,-5)= -3

For node G, max(0, -∞) =>max(0,7) =7



In step 3 :

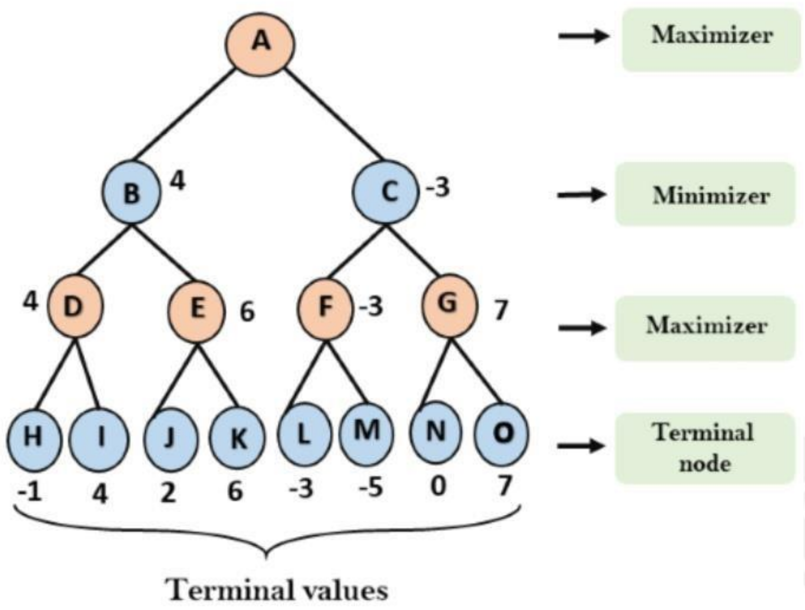
Its turn of minimizer, so it will compare all nodes value with +∞, and will find the 3rd layer node values.

For node B:

min(4,6) =4

For node C:

min(-3,7)= -3



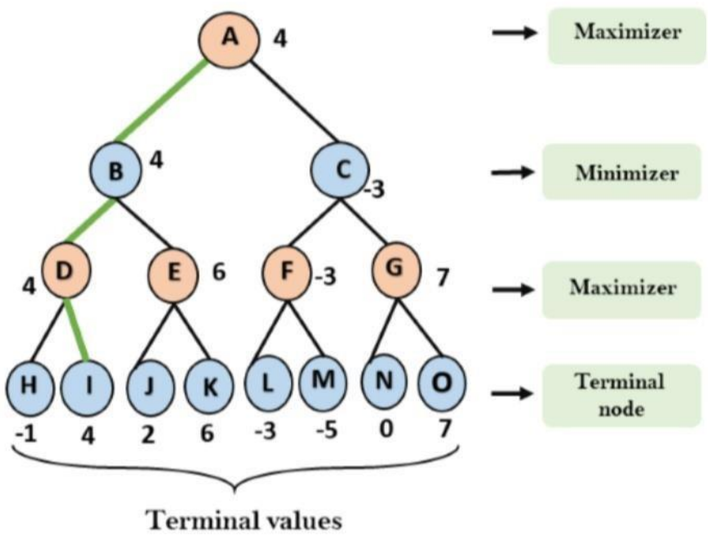
Step 4:

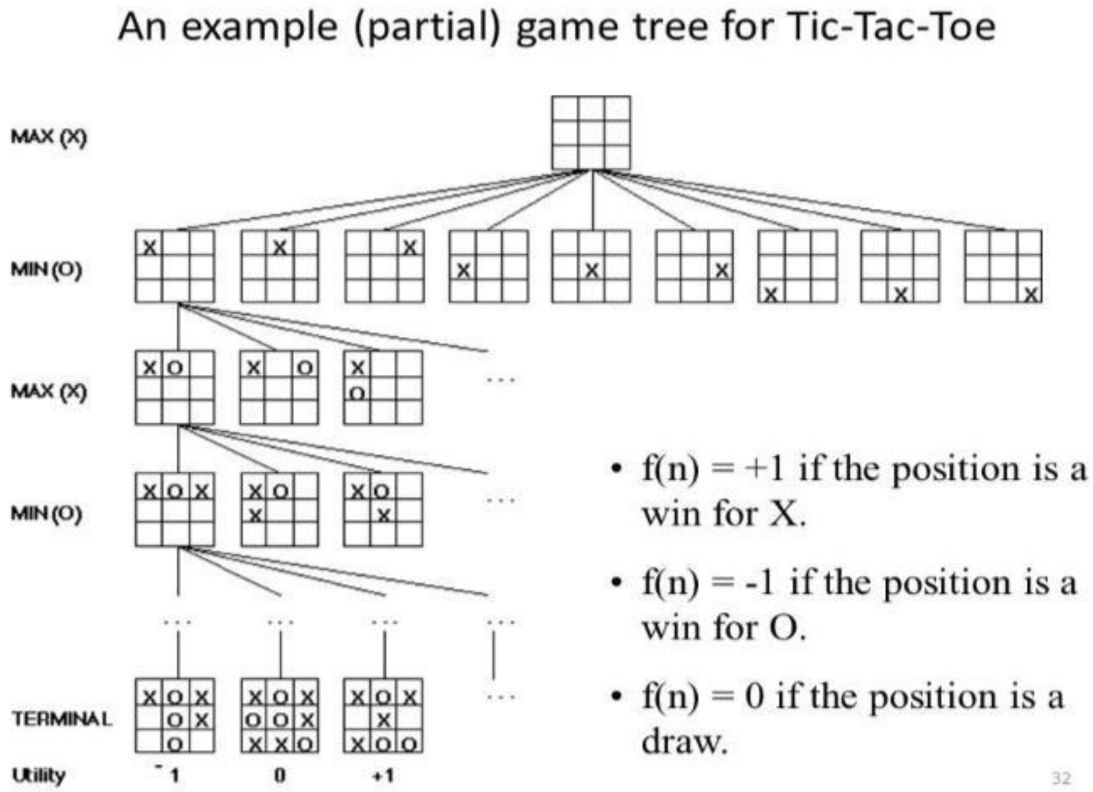
Now it's a turn for maximize, and it will again choose the maximum of all nodes value and find the maximum value for the root node. In this game tree, there are only 4 layers, hence we reach immediately to the root node, but in real games, there will be more than 4 layers.

For node A

max(4,-3)=4

That was the complete work flow of the minimax two player game.




An example (partial) game tree for Tic-Tac-Toe
MAX (X)
MIN (O)
MAX (X)
MIN (O)
TERMINAL
Utility
-1
0
+1
• f(n) = +1 if the position is a win for X.
• f(n) = -1 if the position is a win for O.
• f(n) = 0 if the position is a draw.
32