



STACK IN DS

C PROGRAM TO IMPLEMENT
STACK

ADVERTISEMENT

Mini-Max Algorithm in Artificial Intelligence

ADVERTISEMENT

ADVERTISEMENT

- Mini-max algorithm is a 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.
- Mini-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, go, and various tow-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 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, where MAX will select the maximized value and MIN will select the minimized value.
- The minimax algorithm performs a depth-first search algorithm for the exploration of the complete game tree.
- The minimax algorithm proceeds all the way c
backtrack the tree as the recursion.

Pseudo-code for MinMax Algorithm

```
function minimax(node, depth, maximizingPlayer) is
```

```

if depth == 0 or node is a terminal node then
return static evaluation of node

if MaximizingPlayer then    // for Maximizer Player
maxEva= -infinity
for each child of node do
eva= minimax(child, depth-1, false)
maxEva= max(maxEva,eva)    //gives Maximum of the values
return maxEva

else                        // for Minimizer player
minEva= +infinity
for each child of node do
eva= minimax(child, depth-1, true)
minEva= min(minEva, eva)    //gives minimum of the values
return minEva

```

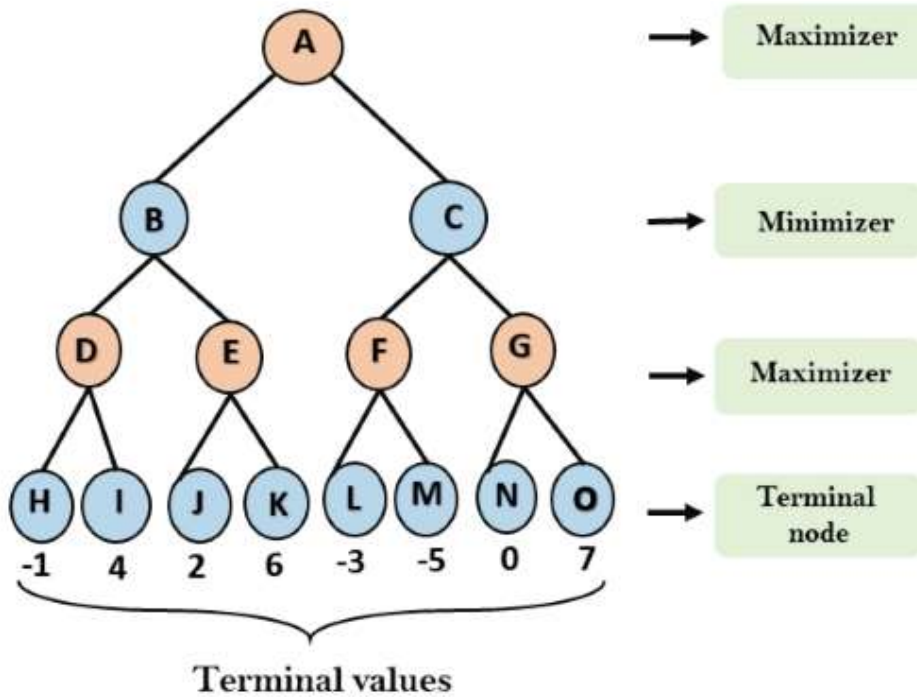
Initial call:

Minimax(node, 3, true)

Working of Min-Max Algorithm:

- The working of the minimax algorithm can be easily described using an example. Below we have taken an example of game-tree which is representing the two-player game.
- In this example, there are two players one is called Maximizer and other is called Minimizer.
- Maximizer will try to get the Maximum possible score, and Minimizer will try to get the minimum possible score.
- This algorithm applies DFS, so in this game-tree, we have to go all the way through the leaves to reach the terminal nodes.
- 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 involved in solving the two-player game tree: ⊗

Step-1: In the first step, the algorithm generates the initial state of the tree to get the utility values for the terminal states. In the state of the tree. Suppose maximizer takes first turn with the best-case initial value and minimizer will take next turn which has worst-case initial value.



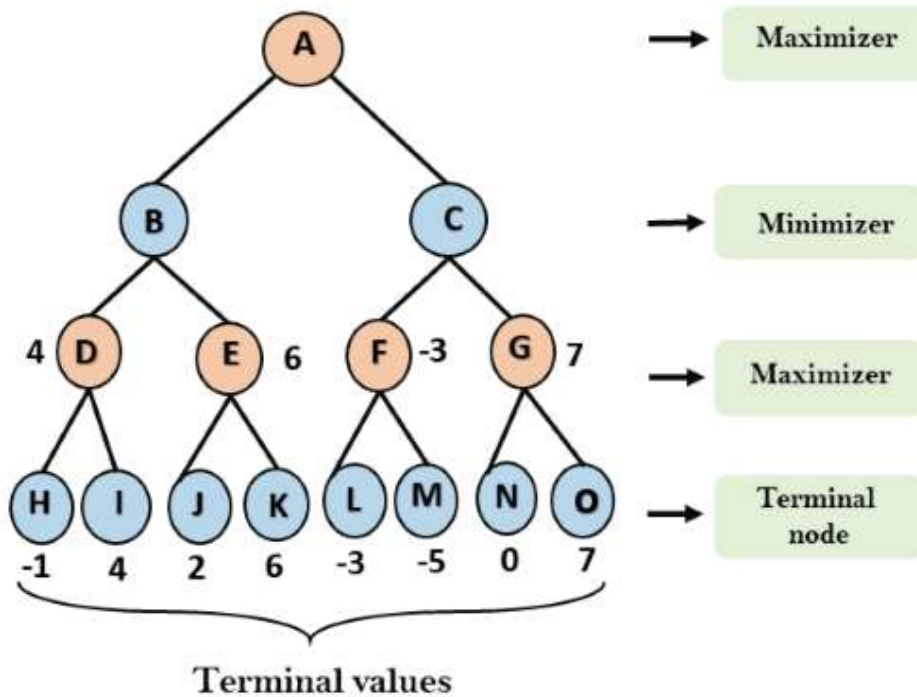
Step 2: Now, first we find the utilities value for the Maximizer, its initial value is $-\infty$, so we will compare each value in terminal state with initial value of Maximizer and determines the higher nodes values. It will find the maximum among the all.

ADVERTISEMENT



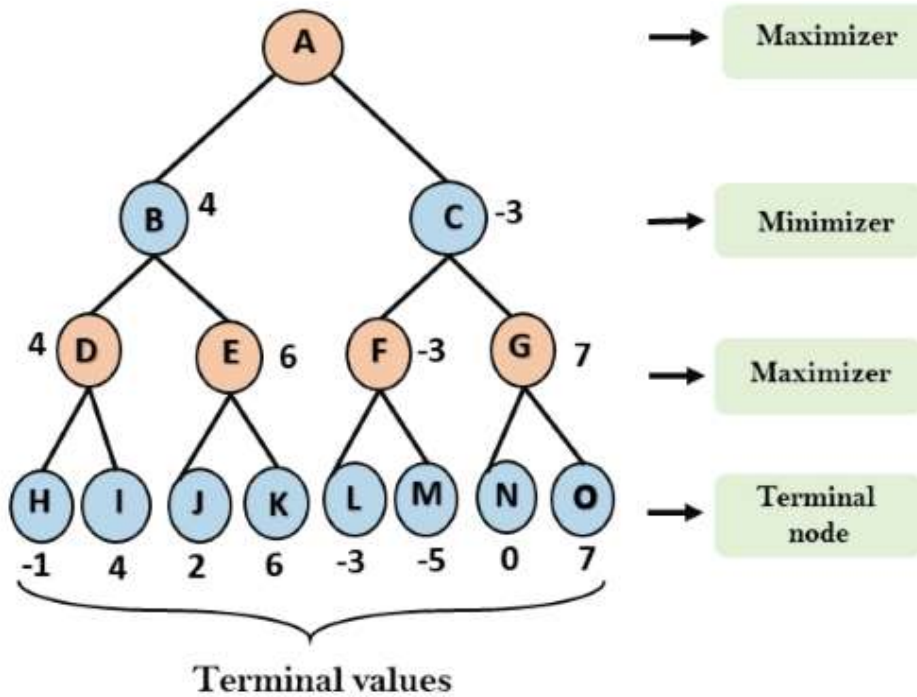
- For node D $\max(-1, -\infty) \Rightarrow \max(-1, 4) = 4$
- For Node E $\max(2, -\infty) \Rightarrow \max(2, 6) = 6$
- For Node F $\max(-3, -\infty) \Rightarrow \max(-3, -5) = -3$

- For node G $\max(0, -\infty) = \max(0, 7) = 7$



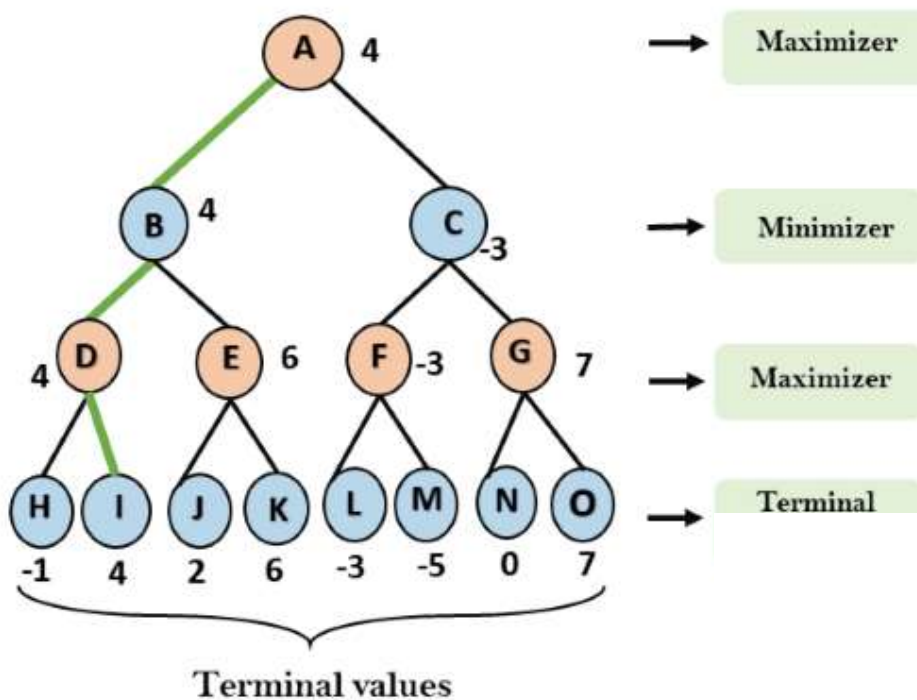
Step 3: In the next step, it's a turn for minimizer, so it will compare all nodes value with $+\infty$, and will find the 3rd layer node values.

- For node B = $\min(4, 6) = 4$
- For node C = $\min(-3, 7) = -3$



Step 4: Now it's a turn for Maximizer, 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 workflow of the minimax two pl

Properties of Mini-Max algorithm:

- **Complete-** Min-Max algorithm is Complete. It will definitely find a solution (if exist), in the finite search tree.
- **Optimal-** Min-Max algorithm is optimal if both opponents are playing optimally.
- **Time complexity-** As it performs DFS for the game-tree, so the time complexity of Min-Max algorithm is $O(b^m)$, where b is branching factor of the game-tree, and m is the maximum depth of the tree.
- **Space Complexity-** Space complexity of Mini-max algorithm is also similar to DFS which is $O(bm)$.

Limitation of the minimax Algorithm:

The main drawback of the minimax algorithm is that it gets really slow for complex games such as Chess, go, etc. This type of games has a huge branching factor, and the player has lots of choices to decide. This limitation of the minimax algorithm can be improved from **alpha-beta pruning** which we have discussed in the next topic.

[< Prev](#)[Next >](#)

ADVERTISEMENT

ADVERTISEMENT



 Youtube **For Videos Join Our Youtube Ch**
















Feedback

- Send your Feedback to feedback@javatpoint.com






Help Others, Please Share



Learn Latest Tutorials

 Splunk tutorial Splunk	 SPSS tutorial SPSS	 Swagger tutorial Swagger	 T-SQL tutorial Transact-SQL
 Tumblr tutorial Tumblr	 React tutorial ReactJS	 Regex tutorial Regex	 Reinforcement learning tutorial Reinforcement Learning
 R Programming tutorial R Programming	 RxJS tutorial RxJS	 React Native tutorial React Native	 Python Design Patterns Python Design Patterns
 Python Pillow tutorial Python Pillow	 Python Turtle tutorial Python Turtle	 Keras tutorial Keras	













Preparation

 Aptitude Aptitude	 Logical Reasoning Reasoning	 Verbal Ability Verbal Ability	 Interview Questions
 Company Interview Questions			



Company Questions


Trending Technologies

 Artificial Intelligence Artificial Intelligence	 AWS Tutorial AWS	 Selenium tutorial Selenium	 Cloud Computing Cloud Computing
 Hadoop tutorial Hadoop	 ReactJS Tutorial ReactJS	 Data Science Tutorial Data Science	 Angular 7 Tutorial Angular 7
 Blockchain Tutorial Blockchain	 Git Tutorial Git	 Machine Learning Tutorial Machine Learning	 DevOps Tutorial DevOps

B.Tech / MCA

 DBMS tutorial DBMS	 Data Structures tutorial Data Structures	 DAA tutorial DAA	 Operating System Operating System
 Computer Network tutorial Computer Network	 Compiler Design tutorial Compiler Design	 Computer Organization and Architecture Computer Organization	 Discrete Mathematics Tutorial Discrete Mathematics
 Ethical Hacking Ethical Hacking	 Computer Graphics Tutorial Computer Graphics	 Software Engineering Software Engineering	 Linux tutorial Linux
 Cyber Security tutorial Cyber Security	 Automata Tutorial Automata	 C++ tutorial C++	






Java tutorial
Java




.Net
Framework
tutorial
.Net



Python tutorial
Python



List of
Programs
Programs



Control
Systems tutorial
Control System



Data Mining
Tutorial
Data Mining



Data
Warehouse
Tutorial
Data Warehouse

ADVERTISEMENT

ADVERTISEMENT

