Name : Sufiyan Khan

RollNo:A12

Batch : A1

Class : T.E. AI-DS

D.O.P. : 01-09-2023

EXPERIMENT NO. 7

Aim : To implement Min-Max Algorithm to study the concept of decision-making algorithm.

Objective : To understand and study one of the decision-making algorithms used in artificial intelligence and optimization to make choices or reach conclusions based on available information and predefined criteria.

Theory : Let us understand more about this algorithm step-by-step,

1. What do you mean by decision making algorithm?

   A decision-making algorithm is a computational process or set of rules used to make choices or reach conclusions based on available information and predefined criteria. Decision-making algorithms are widely used in various fields, including artificial intelligence, data analysis, optimization, and problem-solving. They help automate and optimize decision-making processes in complex and dynamic environments.

2. What is Min-Max Algorithm?

   The Minimax algorithm is a decision-making algorithm used in two-player, zero-sum games, such as chess, checkers, tic-tac-toe, and many other board games. The primary goal of the Minimax algorithm is to determine the best move for a player by minimizing the possible loss (hence "min") while assuming the opponent is trying to maximize their gain (hence "max"). It's a fundamental concept in game theory and artificial intelligence for making strategic decisions in competitive settings.

3. How does the Min-Max Algorithm work?

   Here's how the Minimax algorithm works:

   →Game Tree: Represent the game as a tree, where each level of the tree corresponds to a player's turn, and the branches represent possible moves. The tree extends to a certain depth, typically until the game is won or reaches a predefined depth limit.

   →Evaluation Function: Assign a value or score to terminal states (win, lose, or draw) in the game tree. The evaluation function typically reflects the desirability of the outcome for the player.

→Minimization and Maximization: Starting from the root node (the current game state), the algorithm alternates between minimizing and maximizing players. The minimizing player tries to minimize the evaluation function value, while the maximizing player tries to maximize it.

→Recursion: The algorithm recursively explores the game tree by considering all possible moves and their consequences. For each move, it calculates the value of the game state by minimizing or maximizing the child nodes, depending on the current player.

→Backpropagation: As the algorithm traverses the tree, it propagates the minimum and maximum values up the tree to determine the best move for the root node.

→Best Move: After evaluating all possible moves, the algorithm selects the move that leads to the maximum value at the root node for the maximizing player or the minimum value for the minimizing player, depending on whose turn it is.

4. What are the characteristics of Min-Max Algorithm?

→ Two-Player, Zero-Sum Game
→ Recursive Tree Exploration
→ Optimality Under Perfect Play

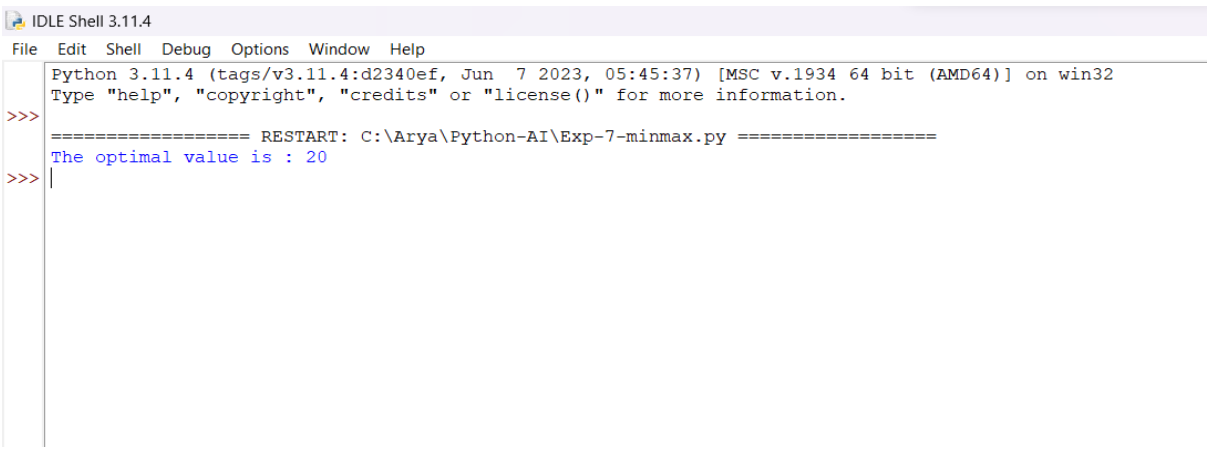
Example :

Program :

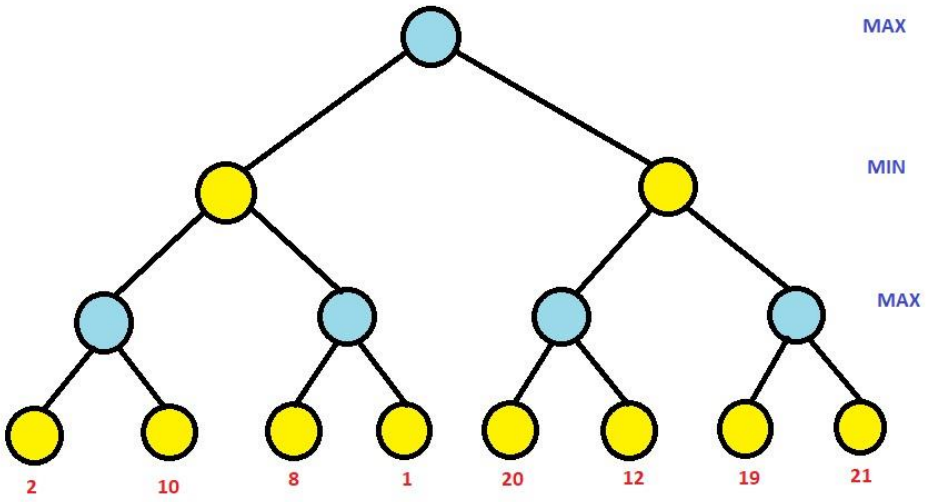
```
import math


def minimax (curDepth, nodeIndex,
            maxTurn, scores,
            targetDepth):



    if (curDepth == targetDepth):
        return scores[nodeIndex]
```

```
    if (maxTurn):
        return max(minimax(curDepth + 1, nodeIndex * 2,
                   False, scores, targetDepth),
                   minimax(curDepth + 1, nodeIndex * 2 + 1,
                   False, scores, targetDepth))

    else:
        return min(minimax(curDepth + 1, nodeIndex * 2,
                   True, scores, targetDepth),
                   minimax(curDepth + 1, nodeIndex * 2 + 1,
                   True, scores, targetDepth))


scores = [2, 10, 8, 1, 20, 12, 19, 21]

treeDepth = math.log(len(scores), 2)

print("The optimal value is : ", end = "")
print(minimax(0, 0, True, scores, treeDepth))
```

Output :



Now, let us see the graph that we get from the input that we have given,



We can see that the terminal nodes have the values that we have given as input now let us see how this is solved,

(i) We start the labelling of max and min level from the root node level, we consider the root node level as maximum value level and the level below it as minimum value level.

(ii) Now, we start the solving from the terminal nodes, consider the first two terminal nodes which are actually the children nodes of the node above it, or to which they are connected (parent node).

(iii) The parent node for the terminal nodes is in the maximum value level so we take the value of the parent node as the value which is maximum between the two children nodes, as the parent node has no value as for yet.

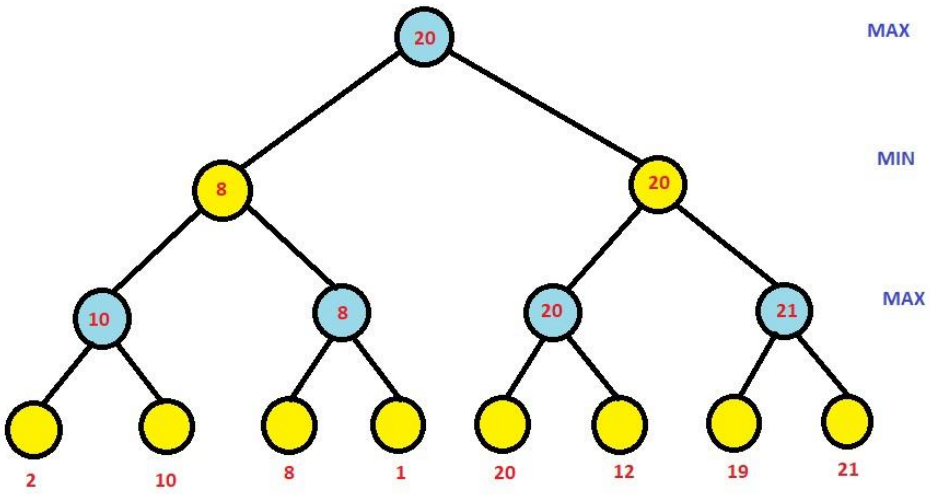
(iv) So here, we take node '2' and node '10' and check which one is maximum and it is very obvious that 10 > 2 so we take the value of their parent node as 10.

(v) We repeat the same step for the rest of the terminal nodes,
8 > 1 = 8
20 > 12 = 20
19 < 21 =21

(vi) Now, the parent nodes in the previous step have become children nodes of the nodes in the minimum level or the level above them, here we take the value of the parent node as the value which is minimum between the two children nodes,
10 < 8 = 8
20 < 21 = 20

(vii) Now for the last step, we take the minimum value level nodes as the children nodes of the root node which is a maximum level node, here we will take the value for the root node/ optimal value which will be maximum between the two children nodes,
8 < 20 = 20

(viii) Therefore we get the optimal value as 20.



This is how we solve a Min-Max Algorithm problem.

CONCLUSION :

Hence in this experiment we studied the concept of decision-making algorithm by considering the Min-Max Algorithm and successfully implemented it. Thus the aim has been satisfied.