Name : Sufiyan Khan

RollNo:A12

Batch : A1

Class : T.E. AI-DS

D.O.P. : 13-10-2023

EXPERIMENT NO. 8

Aim : To implement Alpha-Beta Pruning in Min-Max Algorithm.

Objective : To understand and study one of the decision-making algorithms to reduce the number of nodes evaluated in a game tree.

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 Alpha-Beta Pruning?

   Alpha-beta pruning is an optimization technique used in decision-making algorithms, particularly in the Minimax algorithm, to reduce the number of nodes evaluated in a game tree. It helps improve the efficiency of the search by eliminating branches that cannot influence the final decision.The primary goal of alpha-beta pruning is to identify situations where it becomes clear that certain moves are inferior for the maximizing player (Max) or the minimizing player (Min), without the need to explore the entire subtree. By pruning these unpromising branches, the algorithm can reduce the computational effort required to find the optimal move.

3. How does the Alpha-Beta Pruning work?

   Here's how alpha-beta pruning works:

   →Initialization:

   Alpha is initially set to negative infinity (representing the worst possible value for Max).
   Beta is initially set to positive infinity (representing the worst possible value for Min).
   Traversal:

The Minimax algorithm proceeds to explore the game tree in a depth-first manner, considering possible moves.

→Evaluating Nodes:

As the algorithm evaluates nodes in the tree, it updates the alpha and beta values based on the best outcomes found so far for Max and Min, respectively.

→Pruning Condition:

Alpha-beta pruning comes into play when it's determined that a node's value cannot affect the final decision. For Max nodes, if the current node's value is greater than or equal to beta (i.e., Max already has a better option), the remaining nodes in that branch can be pruned. For Min nodes, if the current node's value is less than or equal to alpha (i.e., Min already has a better option), the branch can be pruned.

→Optimized Search:

By pruning unpromising branches, the search space is significantly reduced. This optimization leads to a more efficient exploration of the game tree.

<u>**Example**</u> :

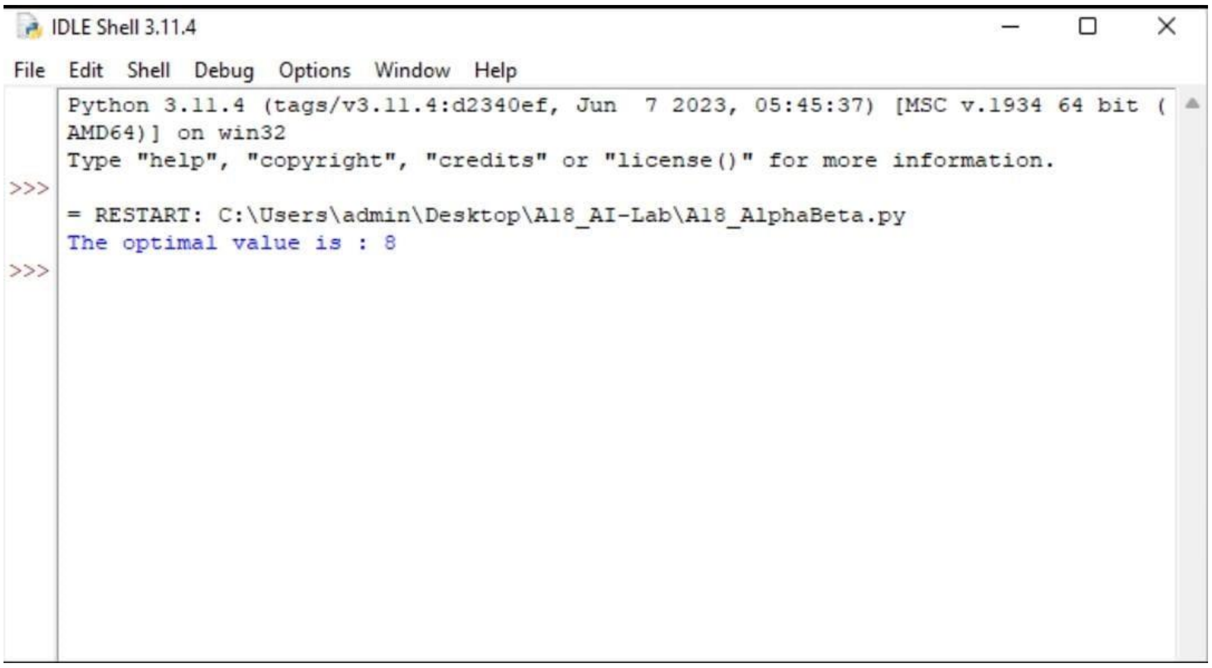
<u>**Program**</u> :

```
MAX, MIN = 1000, -1000


def minimax(depth, nodeIndex, maximizingPlayer,

          values, alpha, beta):



   if depth == 3:

      return values[nodeIndex]


   if maximizingPlayer:


      best = MIN
```

```
    for i in range(0, 2):

        val = minimax(depth + 1, nodeIndex * 2 + i,
                      False, values, alpha, beta)
        best = max(best, val)
        alpha = max(alpha, best)

        if beta <= alpha:
            break

    return best

else:
    best = MAX

    for i in range(0, 2):

        val = minimax(depth + 1, nodeIndex * 2 + i,
                       True, values, alpha, beta)
        best = min(best, val)
        beta = min(beta, best)

        if beta <= alpha:
            break

    return best
```

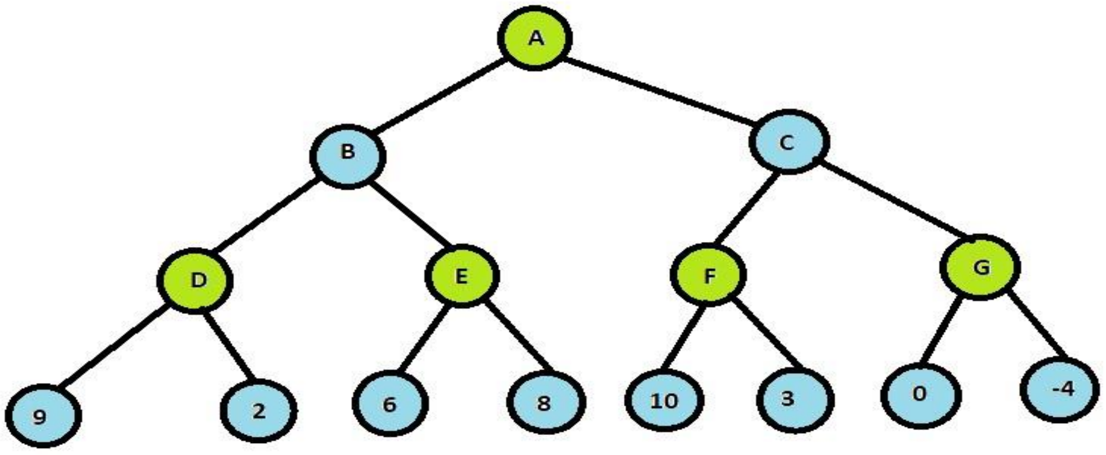
```
if __name__ == "__main__":

    values = [9, 2, 6, 8, 10, 3, 0, -4]
    print("The optimal value is :", minimax(0, 0, True, values, MIN, MAX))
```

Output :



```
Python 3.11.4 (tags/v3.11.4:d2340ef, Jun  7 2023, 05:45:37) [MSC v.1934 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\admin\Desktop\A18_AI-Lab\A18_AlphaBeta.py
The optimal value is : 8
>>>
```

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,

→Root node (A)

Alpha = -∞

Beta = ∞

Child nodes of A:

B (minimizing)

E (maximizing)

Node B

Alpha = -∞

Beta = ∞

Child nodes of B:

C (terminal, utility 8)

D (terminal, utility 9)

Node C

This is a terminal node, so the utility value of 8 is returned.

→Node D

This is a terminal node, so the utility value of 9 is returned.

→Node B (continued)

The utility values of the child nodes of B are 8 and 9. Therefore, beta is updated to the minimum of beta and 8, which is 8.

→Node E

Alpha = -∞

Beta = 8

Child nodes of E:

F (terminal, utility 2)

G (terminal, utility 10)

→Node F

This is a terminal node, so the utility value of 2 is returned.

→Node G

This is a terminal node, so the utility value of 10 is returned.

→Node E (continued)

The utility values of the child nodes of E are 2 and 10. Since beta is less than or equal to alpha, the subtree below node E can be pruned.

→Node A (continued)

Since node E was pruned, the only child node of A that remains is node B. Therefore, the utility value of node B, which is 8, is returned.

Therefore the alpha-beta pruning algorithm returns the value of alpha at the root node, which is 8.

This is how we perform Alpha-Beta Pruning on Min-Max Algorithm.

CONCLUSION :

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