Experiment - 6

Name: - Devansh Shukla
Prn No: - 20200802109
Track: - AI

Aim:

Program to implement, game playing: Minimax, alpha-beta pruning

Software Required:

PyCharm

Theory:

Alpha-beta pruning is an optimization technique used in the minimax algorithm for decision-making in two-player adversarial games, such as chess, checkers, or tic-tac-toe. The minimax algorithm aims to find the best move for a player by simulating all possible moves and their outcomes. The alpha-beta pruning technique significantly reduces the number of nodes evaluated in the minimax search tree by eliminating irrelevant subtrees, thus speeding up the search without affecting the final decision.

Algorithm:

```python
import math

MAX, MIN = math.inf, -math.inf
values = [8, 6, 1, 9, 3, 7, 2, 5]


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


print("The optimal value is :", minimax(0, 0, True, values, MIN, MAX))
```

Applications:

Alpha-beta pruning is widely used in chess, checkers, tic-tac-toe, and other similar board games to efficiently search through the vast number of possible game states and select optimal moves. In video game AI, alpha-beta pruning is utilized to enhance the decision-making capabilities of non-player characters in games like strategy games, racing games, and more. Alpha-beta pruning is a crucial optimization technique that significantly reduces the search space in adversarial games, making it feasible to perform deep searches and find optimal moves in complex, branching-factor-heavy game scenarios.

Output: