Name: Shreyash Gajanan Nale.
Class: TY IT A
Roll No: 331002
Batch: A1
Title: *Implement Best-First Search (Best-Solution but not always optimal)*

Aim:

To study and implement the Min-Max search algorithm with Alpha-Beta Pruning, a heuristic game tree search strategy, and understand how it improves efficiency in decision-making for two-player games by reducing the number of nodes explored. The assignment also aims to analyze its working, advantages, drawbacks, and applications in solving competitive game problems.

Theory:

Min-Max Search with Alpha-Beta Pruning:

The Min-Max algorithm is a recursive search strategy used in decision-making and game theory for two-player games such as Chess or Tic-Tac-Toe. One player is the **Maximizer**, aiming to maximize the score, while the other is the **Minimizer**, aiming to minimize it.

- **Alpha-Beta Pruning:** An optimization technique for Min-Max that eliminates branches in the search tree which do not affect the final decision, thus reducing the number of nodes evaluated.
- **Traversal Pattern:** Performs a depth-first traversal of the game tree, applying pruning when possible.
- **Approach Type:** Heuristic-driven and adversarial search.
- **Memory Requirements:** Requires recursion stack and game tree storage; pruning reduces unnecessary expansion.
- **Drawbacks:** Performance depends on depth limit and heuristic evaluation quality.

Complexity Analysis:

Algorithm	Time Complexity	Space Complexity	Notes
Min-Max (without pruning)	O(b^m)	O(bm)	b = branching factor, m = maximum depth.
Min-Max with Alpha-Beta Pruning	O(b^(m/2))	O(bm)	Efficiency improves with good move ordering

Applications:

- **Board Games:** Widely used in Chess, Tic-Tac-Toe, Checkers for move prediction.
- **AI in Gaming:** Used in adversarial AI agents for decision-making.
- **Robotics and Planning:** Applied in environments involving competitive strategy.
- **Problem Solving:** Useful where two entities compete with opposite goals.

Example and Algorithm:

Implementing Min-Max with Alpha-Beta Pruning for a simplified two-player game.

Min-Max with Alpha-Beta Pruning Pseudocode:

```
function minimax(node, depth, alpha, beta, maximizingPlayer):
    if depth == 0 or node is terminal:
        return heuristic value of node
```

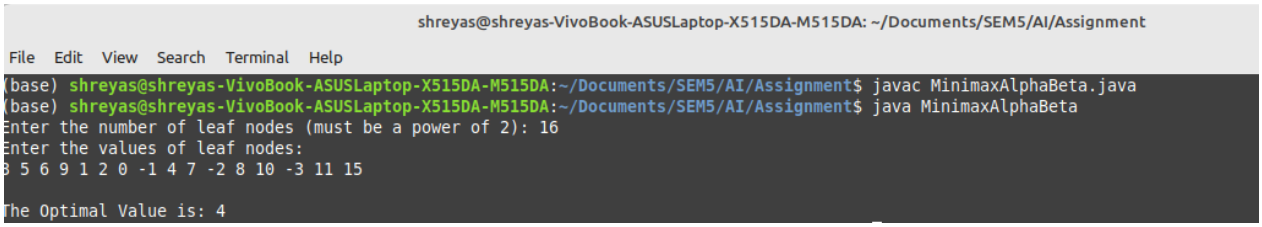
```
    if maximizingPlayer:
        maxEval = -∞
        for each child of node:
            eval = minimax(child, depth - 1, alpha, beta, false)
            maxEval = max(maxEval, eval)
            alpha = max(alpha, eval)
            if beta <= alpha:
                break  // β cut-off
        return maxEval
    else:
        minEval = +∞
        for each child of node:
            eval = minimax(child, depth - 1, alpha, beta, true)
            minEval = min(minEval, eval)
            beta = min(beta, eval)
            if beta <= alpha:
                break  // α cut-off
        return minEval
```

Code And Output:

```
import java.util.*;


public class MinimaxAlphaBeta {
    static int minimax(int depth, int nodeIndex, boolean isMax, List<Integer> scores, int alpha, int beta, int maxDepth) {
        if (depth == maxDepth) return scores.get(nodeIndex);
```

```
    if (isMax) {
        int best = -1000;
        for (int i = 0; i < 2; i++) {
            int val = minimax(depth + 1, nodeIndex * 2 + i, false, scores, alpha, beta, maxDepth);
            best = Math.max(best, val);
            alpha = Math.max(alpha, best);
            if (beta <= alpha) break;
        }
        return best;
    } else {
        int best = 1000;
        for (int i = 0; i < 2; i++) {
            int val = minimax(depth + 1, nodeIndex * 2 + i, true, scores, alpha, beta, maxDepth);
            best = Math.min(best, val);
            beta = Math.min(beta, best);
            if (beta <= alpha) break;
        }
        return best;
    }
}

public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    System.out.print("Enter the number of leaf nodes (must be a power of 2): ");
    int numLeaves = sc.nextInt();

    List<Integer> scores = new ArrayList<>();
    System.out.println("Enter the values of leaf nodes:");
```

```
        for (int i = 0; i < numLeaves; i++) {
            scores.add(sc.nextInt());
        }

        int maxDepth = 0, temp = numLeaves;
        while (temp > 1) {
            temp /= 2;
            maxDepth++;
        }

        int optimalValue = minimax(0, 0, true, scores, -1000, 1000, maxDepth);
        System.out.println("\nThe Optimal Value is: " + optimalValue);
    }
}
```



Conclusion:

Min-Max with Alpha-Beta Pruning is a powerful adversarial search method used in two-player games. It provides optimal decisions while significantly reducing the number of nodes explored compared to plain Min-Max. Though its efficiency depends on move ordering and heuristic evaluation, its application in game AI, robotics, and strategic decision-making makes it a cornerstone algorithm in Artificial Intelligence.