Assignement 2

Team : Arsany Osama 2205122

Mark Magdy 2205040

Ahmed Abdelrahman 2205008

Comparison of Minimax Algorithms with and without Alpha-Beta Pruning

1. Introduction:

In this report, we compare two versions of the Minimax algorithm for a game (Connect Four):

one with alpha-beta pruning and one without alpha-beta pruning. We analyze the performance of both algorithms in terms of time taken and nodes expanded at different depth levels (K values). We also provide sample runs with corresponding minimax trees to illustrate the differences between the two algorithms.

2. Assumptions:

- We assume that the game is a two-player, turn-based, zero-sum game.
- The game board is represented as a 2D array.
- The evaluation function assigns scores to different board states based on the number of Connect Fours and potential Connect Fours for each player.

3. Data Structures and Algorithms:

- Both algorithms use a recursive approach to explore the game tree.
- The Minimax algorithm without alpha-beta pruning uses a simple depth-first search to evaluate all possible moves.
- The Minimax algorithm with alpha-beta pruning optimizes the search by pruning branches that are guaranteed to be worse than previously explored branches. It uses alpha and beta values to keep track of the best scores found so far.

4. Sample Runs and Minimax Trees:

Here, we provide two sample runs for a Connect Four game with a 6x7 board size. The 'Y' player represents the AI player, and the 'R' player represents the opponent.

Sample Run 1 (Depth: 3):

Initial Board State:

```
  0  1 2  3  4 5  6
{' ', ' ', ' ', ' ', ' ', ' ',' ' },
{' ', ' ', ' ', ' ', ' ', ' ',' ' },
{' ', ' ', ' ', ' ', ' ', ' ',' ' },
{' ', ' ',  'R', ' ', ' ', ' ',' ' },
{'Y', ' ', 'R', ' ', ' ', ' ',' '},
{'Y', ' ', 'R', ' ', ' ', ' ',' '}
```

Minimax Tree (without Alpha-Beta Pruning):

Max (2)

Min (2) Min (0) Min (0)

Max Max Max Max Max Max Max Max Max

(4) (2) (3) (0) (6) (1) (5) (2) (0)

After using evaluate method the ai will choose to play (Y) in column with index : 2

<u>Sample Run 2 (Depth: 3):</u>

Initial Board State:

```
 0  1  2  3  4  5  6
{' ', ' ', ' ', ' ', ' ', ' ',' ' },
{' ', ' ', ' ', ' ', ' ', ' ',' ' },
{' ', ' ', ' ', ' ', ' ', ' ',' ' },
{' ', ' ', 'R', ' ', ' ', ' ',' ' },
{'Y', ' ', 'R', ' ', ' ', ' ',' '},
{'Y', ' ', 'R', ' ', ' ', ' ',' '}
```

Minimax Tree (with Alpha-Beta Pruning):

Max (2)

Min (2) Min (0) Min (2)

Max Max Max(Prune) Max Max(Prune) Max(Prune) Max Max Max(Prune)

(4) (2) (3) (0) (6) (1) (5) (2) (0)

The ai will choose column 2 according to the values of Alpha-Beta and if Alpha>=Beta all the following nodes will delete them and by this the algorithm will take less time reaching the max. or best solution according to depth

<u>5. Performance Comparison:</u>

We compare the two algorithms in terms of time taken and nodes expanded at different depth levels (K values). The performance metrics are recorded for multiple runs and averaged to obtain reliable results.

Let’s see sample data comparison of k values ={1,2,3,4}

At Depth (K),Time Taken (Alpha-Beta), Nodes Expanded (Alpha-Beta),Time Taken (No Alpha-Beta), Nodes Expanded (No Alpha-Beta)

At Depth (K)	Time Taken (Alpha-Beta)	Nodes Expanded (Alpha-Beta)	Time Taken (No Alpha-Beta)	Nodes Expanded (No Alpha-Beta)
K = 1	1 ms	56 nodes	1 ms	56 nodes
K = 2	2 ms	227 nodes	3 ms	399 nodes
K = 3	4 ms	1248 nodes	6 ms	2800 nodes
K = 4	8 ms	4602 nodes	24 ms	19607 nodes

<u>Important note:</u> these data as a sample it may be change according to the human play

6. Conclusion:

Based on the analysis, we observe that the Minimax algorithm with alpha-beta pruning outperforms the version without alpha-beta pruning in terms of time taken and nodes expanded. The use of alpha-beta pruning allows for the pruning of branches that are guaranteed to be worse, resulting in a more efficient search and faster decision-making. As the depth level increases, the difference in performance between the two algorithms becomes more significant.

However, it's important to note that the actual time taken and nodes expanded may vary depending on the specific game and implementation details. The performance comparison provided in this report serves as a general illustration of the advantages of alpha-beta pruning.

To further enhance the performance, additional optimizations can be considered, such as move ordering and transposition tables, which can reduce redundant evaluations and improve the effectiveness of alpha-beta pruning.

Overall, the Minimax algorithm with alpha-beta pruning is a powerful technique for decision-making in two-player games, providing a balance between accuracy and efficiency.