|  |
| --- |
| Måns Alklint  2024-01-24 |

|  |
| --- |
| LTH | Computer science |
| EDAP01 Report |
| Assignment 1 |

Index

[Approach 2](#_Toc158170318)

[Launching the application 3](#_Toc158170319)

[3 Peer-review (1-2 pages) 4](#_Toc158170320)

[Interaction summary with an AI chatbot (1-2 pages) 5](#_Toc158170321)

# Approach

Minimax Algorithm

The minimax algorithm is a decision-making algorithm used for games with two players, where the goal is to find the optimal move for the current player while assuming that the opponent plays optimally as well. It works by recursively exploring the game tree, alternating between maximizing and minimizing the player's scores at each level of the tree until a terminal state (win/lose/draw) or a maximum depth is reached.

Alpha-Beta Pruning

Alpha-beta pruning is an optimization technique applied to the minimax algorithm to reduce the number of nodes evaluated in the game tree. It maintains two values, alpha and beta, which represent the minimum score the maximizing player is assured of and the maximum score the minimizing player is assured of, respectively. During the minimax search, if it is found that a certain branch does not affect the final result, it can be pruned by cutting off further exploration along that branch.

Function Minimax

This function defines the function responsible for the search algorithm to iterate through possible moves in the connect-4 game. The function works by recursion, which in short is a sort of a loop. The loop breaks as the search depth in the search tree reaches 0 or if the game is finished (i.e is win, tie or loss). The alpha and beta values during the search cuts off branches that are deemed irrelevant based on the current alpha and beta values.

Function evaluation

The minimax-function uses a helper method to evaluate (give each node in the tree a stochastic value), as this is needed for determining if the move is considered good or bad. It assigns infinity if the player has won, negative infinity if the opponent has won, and computes a score based on the positions of the pieces on the board. This score is a preset represented as a shadowed matrix where each cell has its respective “score”-value.

Function “winning move”

The evaluation needs this small helper method to check the board for positions that gives the respective player’s marker on the board its “score” (the stochastic value). For instance, if a player has a set of markers in a line with 3 as length, the next marker to be placed is deemed a “winning move”. This is done using loops that checks the board horizontally, vertically and askew.

# Launching the application

The application’s perquisites can be installed automatically using the following command:

pip install -r requirements.txt

Then the help commands for executing the application can be found using:

python game.py -h

# 3 Peer-review (1-2 pages)

Name your partner and the points you discussed.

3.1 Peer’s Solution

Describe the solution of your peer

3.2 Technical Differences of the Solutions

Which things were done differently?

3.3 Opinion and Performance

• Which differences are most important?

• How does one assess the performance?

• Which solution would perform better?

1

# Interaction summary with an AI chatbot (1-2 pages)

4.1 Interaction prompts

The prompts that made the chatbot to produce code that could run the 4inarow

game against the server.

4.2 Difference to the own solution

Elaborate on the differences of the obtained code to your solution.

4.3 Performance

Is there any substantial difference of performance between your and chatbot’s

solutions?

2