|  |
| --- |
| Close-up image showing the leaf-sides of two oversized books side-by-side on a bookshelf, with additional books in soft focus background |
| Tic-Tac-Toe game  Coursework 3 |
| |  |  |  | | --- | --- | --- | | Tony, YIT19488399 | 4/19/22 | Artificial Intelligence | |

• “A brief explanation of the strategy or algorithm that you use to determine the move for AI player.”

The algorithm chosen in my Tic-Tac-Toe game is MINIMAX because of Lab week 6 on the AI module. This is a great opportunity to apply a minimax search algorithm to this game with two players: MAX and MIN.

Max is maximizing score value in each move and Min minimizes score value in each move. If both players play optimally, it will be a draw at the end of the game. Game tree representation is complex and large to draw. See the link for a better understanding of the tree. <https://minimax-visualizer.herokuapp.com/>. Basically, minimax determines the optimal strategy and movement for each player every turn, and therefore it is a search algorithm. It checks all the possible positions on the game board in each movement and decides the best one independently for each player.

Minimax is a recursive algorithm. Also, can be a backtracking algorithm from the terminal state or node.

AI player looks intelligent and smart, but it is code implemented especially. Correct indentations with right loops and if statements are the secret of this simple code. Nothing intelligent! Only lines of code! Human NEVER can win against an AI player, called robot in my code. The best end in this game for the human player is a draw.

I included a timer countdown in my code for entertainment purposes only. AI player moves much faster than a human player. And if the human player needs more time to choose a cell, this is ok, and the game will continue.

• Result of a demo run of the game. You should take screenshots of your results from Jupyter Notebooks and insert them into your report.

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

• A brief reflection. It may include but is not limited to the discussions about the topics below:

o Have you met any difficulties/bugs during this coursework, and how did you tackle them?

The most difficult part was designing the minimax algorithm because it uses concepts like a tree, recursion, node, score, etc.

Another difficulty found was the design of the game table. In the beginning, I tried to make a coloured table script executing in a new window. This is difficult in Jupyter Notebook, and I just printed it in black and white on the console at the end.

o What did you think is the most difficult part of this coursework?

It is impossible to win the robot! Human player can only aspire to the draw. Without the Minimax algorithm, the robot sometimes chooses not optimal cells of the board and the human player can win. But this is not the case with this Tic Tac Toe game.

o What have you learnt from investigating this problem?

I learned about the minimax algorithm, decision tree and other algorithms like Alpha-Beta Pruning, but I did not use it here.

And I found an interesting Tic Tac Toe game on the website. It is called the “Ultimate Tic Tac Toe” game. <https://ultimate-t3.herokuapp.com/>. It is much harder to code it and play it.

o Which part(s) in your code design is your favourite and why?

My favourite part is the timer because it makes the game more interesting and challenging. It remembers me of chess playing between two human players in the past and I applied the timer concept to the Tic Tac Toe game.

o Which of the searching algorithms that you implemented is the best? Why?

There is only one algorithm, minimax. Reasons are given previously in this report. Another interesting algorithm that I could implement is “Alpha-Beta Pruning”. This uses a heuristic value in the minimax algorithm to discard movements that are worse than the previously examined move. It is an algorithm to improve the minimax algorithm. <https://stackabuse.com/minimax-and-alpha-beta-pruning-in-python/>.

o Any references you might have used to complete the task.

Wikipedia, AI Lecture week 6, YouTube and Google. Links are provided previously in this report.

##########################END###########################