Name: Mai Tien Dat Tran
ID: 104207944

GOALS:
"This lab's goal is to provide you with experience representing games as distinct states. Using a graph of states, we may apply actions to investigate a game's conclusion and use graph-based search to find game results.To the right is an example of a tic-tactoe game graph or game tree. Every state that the game could take is a node or vertex on the tree, and every move is an edge that allows us to get from one node to another.Then, our AIs can assess moves by looking through the tree to determine possible paths taken by themselves and by opponents."

TECHNOLOGIES, TOOLS, AND RESOURCES USED:

Tools:

Python 3.12..0

ChatGPT. Retrieved from: chat..openai..com

Resources:

Lab 03 - Tic-Tac-Toe

How to make your Tic Tac Toe game unbeatable by using the minimax algorithm. Retrieved from: https://www.freecodecamp.org/news/how-to-make-your-tic-tac-toe-game-unbeatable-by-using-the-minimax-algorithm-9d690bad4b37/

TASKS UNDERTAKEN:

1.
For the third AI bot, we're implementing a recursive method to randomly search for moves in the graph. The aim is to find the maximum value representing the ThirdBot's victory. It's essentially a simplified version of the minimax algorithm. Each valid move of the bot is followed by a random move of the opponent. The maximum evaluation is returned based on the outcome: ThirdBot wins (1); Opponent win (-1); Tie (0). No pruning method is applied here.

2. Moving on to the fourth AI bot, we're focusing on efficient search, which is a simpler form of Minimax. It evaluates each valid move of the bot and assumes the opponent will do the same. Instead of choosing the maximum value beneficial for the Fourth Bot, the opponent selects the minimum value. The same outcome rules apply: FourthBot wins (1); Opponent win (-1); Tie (0).

3.
Now, for the fifth AI bot, we're enhancing the search efficiency by implementing Alpha-Beta pruning. This involves adding two extra parameters in the recursive method: "alpha" and "beta". "Alpha" represents the best value that the FifthBot, aiming to maximize the outcome, can currently achieve. "Beta" represents the best value that FifthBot's opponent, aiming to minimize the outcome, can achieve. During the search through possible moves of the FifthBot, if a move's evaluation value exceeds the current "alpha", it updates "alpha". Then, it compares the updated "alpha" with the "beta" value calculated in the previous node. If "beta" is not greater than "alpha", it stops searching further moves for the FifthBot, as they would only lead to worse outcomes.

WHAT I FOUND OUT:

Given how obviously "stupid" the third AI bot is, whenever I let it play against the fourth or fifth bot, it typically lost.

When it comes to the minimax algorithm with and without alpha-beta pruning (implemented in the fourth and fifth bots), there is a significant difference in performance: Random Search Bot vs Minimax Bot (without pruning):
First Match: Minimax Bot wins in 0.53 seconds
Second Match: Minimax Bot wins in 0.478 seconds
Third Match: Minimax Bot wins in 0.505 seconds
Random Search Bot vs Minimax Bot (with alpha-beta pruning):
First Match: Minimax Bot (with pruning) wins in 0.236 seconds
Second Match: Minimax Bot (with pruning) wins in 0.237 seconds
Third Match: Minimax Bot (with pruning) wins in 0.22 seconds ...
(The results have been uploaded with the 25 March Commit)
It can be easily seen that alpha-beta prunning has optimized the running time of the Minimax algorithm by more than half.

OPEN ISSUES/RISKS:

Ignoring edge cases or technical errors that could impair the effectiveness of the AI bots are examples of potential dangers.

RECOMMENDATIONS:

With my effort, I've made some recommendations that could be used for the upcoming extension tasks:To increase the effectiveness and efficiency of the AI bot(s), think about adding new features and optimizations. For example, you might look into using other search methods or developing more complex evaluation functions.To make sure the AI bots are reliable and robust in a variety of gaming circumstances, carry out more testing and validation.Keep an eye on the AI bot implementations and make necessary adjustments to handle any new problems or difficulties as they arise.