Artificial Intelligence

Semester Project
Level 3 - Semester 1
2023 / 2024

Team Members

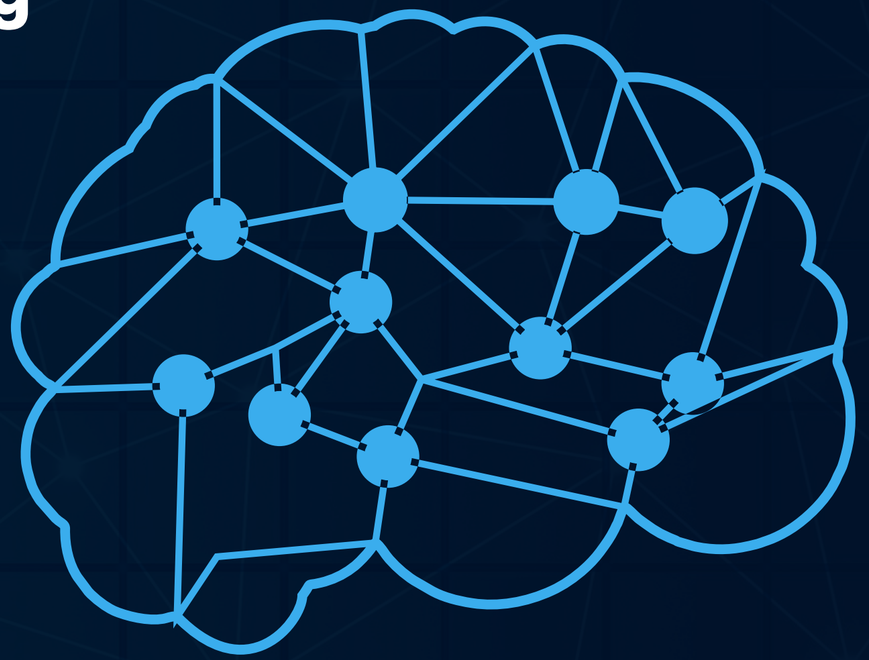
- Omar Khaled Mohamed Sobaih (21-02159)
- Abd El-Kareem Ehab Abd El-Kareem Abd El-Haleem (21-00266)
- Ahmed Shahta Abd El-Azeem (21-00427)
- Youssef Medhat Shawky (21-01732)
- Youssef Mohamed Youssef Emam (21-01750)

Tic-Tac-Toe

Also known as Noughts and Crosses or Xs and Os, is a game for two players.
The players take turns marking the spaces in a three-by-three grid with their chosen symbol, either an ‘X’ or an ‘O’.
The objective of the game is to be the first player to get three of their marks in a row, either horizontally, vertically, or diagonally

The game is implemented in Python, Using Minimax Algorithm

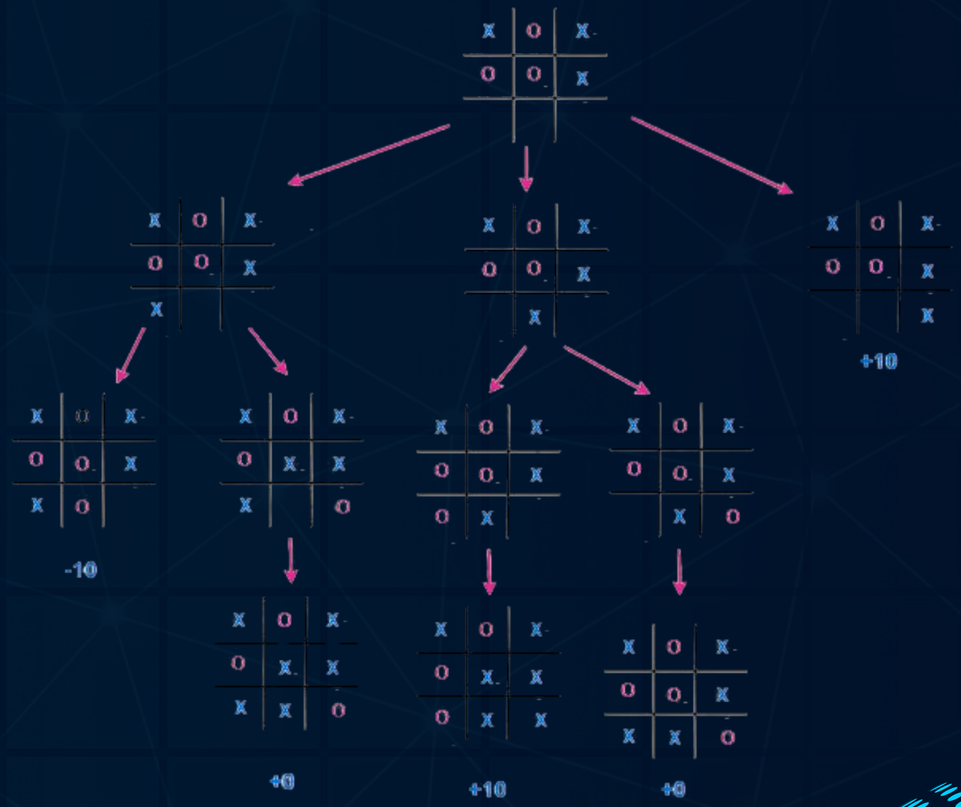
The Minimax algorithm is a decision-making algorithm used in artificial intelligence, decision theory, game theory, statistics, and philosophy for minimizing the possible loss for a worst-case (maximum loss) scenario.
It’s a kind of backtracking algorithm that is used to find the optimal move for a player, assuming that the opponent also plays optimally.



The algorithm works by constructing a game tree of possible moves and outcomes, and then exploring this tree to determine the best move.

It uses a recursive process to evaluate each move:

- First, it generates all possible moves for the current player.
- Then, for each move, it generates all possible responses by the opponent.
- This process continues until it reaches a terminal state (a state where the game ends), at which point it evaluates the outcome of the game.
- The algorithm then backtracks, using the evaluations of the terminal states to evaluate the preceding states, all the way back to the initial state.



The input X / Y coordinates for each cell on the board is assigned as follows :

(0, 0) | (0, 1) | (0, 2)

(1, 0) | (1, 1) | (1, 2)

(2, 0) | (2, 1) | (2, 2)

The user must type the coordinates of the desired cell to take a step on,using only integers.

Code Implementation :

import time module : Provides various time-related functions. It's used in this code to measure the time it takes for the AI agent to make a move.

```
def _init_(self):
    self.initialize_game()
```

This is the constructor of the Game class. It's automatically called when you create a new Game object. It calls the initialize_game method to set up the game.

```
  def initialize_game(self):
      self.current_state = [['_' for _ in range(3)] for _ in range(3)]
      self.player_turn = input("Choose your symbol (X/O):
").upper()
```

This method initializes the game.
It creates a 3x3 grid (represented as a list of lists) for the Tic Tac Toe board, and asks the player to choose their symbol ('X' or 'O').

```
def draw_board(self):
    for i in range(0, 3):
        for j in range(0, 3):
            print('{}|'.format(self.current_state[i][j]), end=" ")
        print()
    print()
```

This method prints the current state of the board.
It uses nested loops to iterate over each cell in the 3x3 grid and print its value.

Code Implementation :

```
def is_valid(self, px, py):
    if px < 0 or px > 2 or py < 0 or py > 2:
        return False
    elif self.current_state[px][py] != '_':
        return False
    else:
        return True
```

This method checks if a move is valid. A move is valid if the chosen cell is within the bounds of the board and is currently empty.

```
def is_end(self):
```

This method checks if the game has ended. The game ends when one player has three of their symbols in a row, column, or diagonal, or when all cells on the board are filled.

```
def max_alpha_beta(self, alpha, beta):
```

This method implements the maximizer part of the Minimax algorithm with Alpha-Beta pruning. It tries to maximize the score by exploring all possible moves and choosing the one with the highest score.

Code Implementation :

```
def min_alpha_beta(self, alpha, beta):
```

This method implements the minimizer part of the Minimax algorithm with Alpha-Beta pruning. It tries to minimize the score by exploring all possible moves and choosing the one with the lowest score.

```
def play(self):
```

This method controls the game flow. It alternates between the player's and the AI's turns, updates the board after each move, and checks if the game has ended.

```
def main():
    g = Game()
    g.play()
```

This function creates a new Game object and starts the game.

```
def main():
    g = Game()
    g.play()
```

This is the entry point of the program. If the script is run directly (not imported as a module), it calls the main function to start the game.

Thank You!

#Free_Palastine