République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université de Saad Dahleb Blida -1-
Faculté des Science
Département d'informatique

REPORT

Option : Master Ingénierie des Systèmes Intelligents (ISI)

Project: Building Intelligent TicTacToe game using python

realized by :

- TEFFAHI Salaheddine
- SLAMANI Abdelmalek

INTRODUCTION:

To solve games using AI, we will introduce the concept of a game tree followed by MINIMAX algorithm. The different states of the game are represented by nodes in the game tree, very similar to the above planning problems. The idea is just slightly different. In the game tree, the nodes are arranged in levels that correspond to each player's turns in the game so that the “root” node of the tree (usually depicted at the top of the diagram) is the beginning position in the game. In tic-tac-toe, this would be the empty grid with no Xs or Os played yet. Under root, on the second level, there are the possible states that can result from the first player’s moves, be it X or O. We call these nodes the “children” of the root node.

Each node on the second level, would further have as its children nodes the states that can be reached from it by the opposing player's moves. This is continued, level by level, until reaching states where the game is over. In tic-tac-toe, this means that either one of the players gets a line of three and wins, or the board is full and the game ends in a tie.

What is Minimax?

The minimax algorithm is a decision-making algorithm commonly used in two-player turn-based games with perfect information, such as Tic Tac Toe, chess, or checkers. The goal of the algorithm is to find the optimal move for a player by evaluating possible moves and choosing the one that leads to the best outcome, assuming the opponent also plays optimally.

Here are the key details about the minimax algorithm:

Objective:

The primary objective of minimax is to determine the best move for the current player in a game with alternating turns.

Maximizer and Minimizer:

- The players are typically referred to as the "maximizer" and the "minimizer."
- The maximizer seeks to maximize the score (favorable outcome) for the current player.
- The minimizer seeks to minimize the score (unfavorable outcome) for the opponent.

Recursive Evaluation:

- The algorithm evaluates each possible move recursively, assigning a score to each game state.
- The score is determined by a utility function that assesses the desirability of a particular game state.

Base Cases:

- The recursion stops when the algorithm reaches a terminal state (win, lose, or draw) or a predefined depth limit in the game tree.

Game Tree:

- The algorithm explores the entire game tree, representing all possible moves and resulting game states, up to a certain depth.