

Spike: 04**Title: Graphs, Search & Rules**

Author: Marco Giacoppo, 104071453

Goals / deliverables:

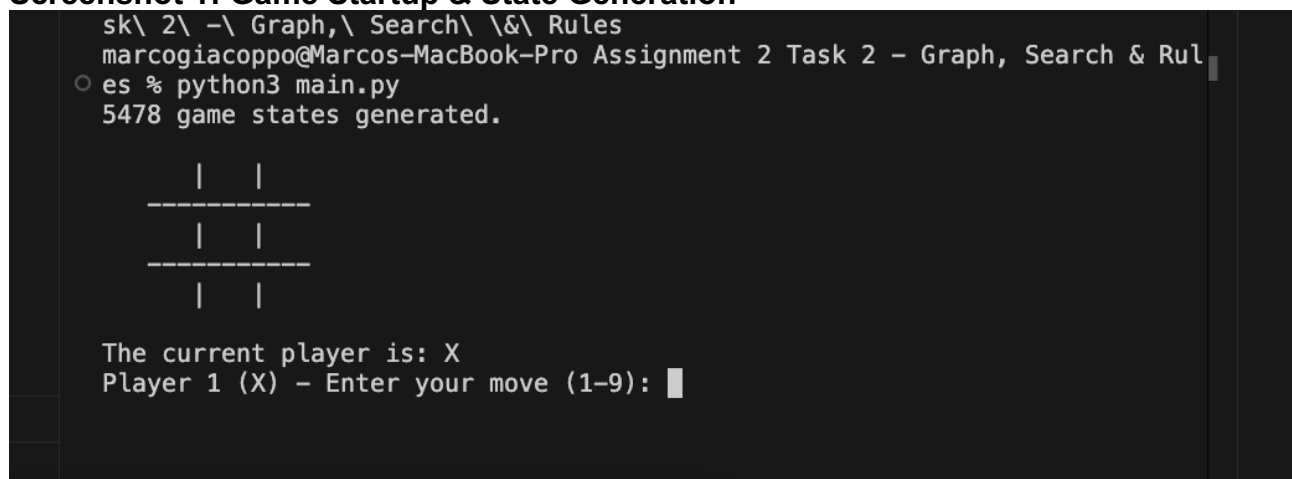
1. Tic-tac-toe code modified to represent the game state as a graph.
2. An AI that randomly searches the graph
3. An AI that improves the efficiency of a basic random search
4. An AI that improves the effectiveness of a basic random search

Technologies, Tools, and Resources used:

- Python 3.11
- Visual Studio Code
- AI strategy planning

Tasks undertaken:

- Designed the software architecture on paper
- Created board representation and render function
- Implemented the game loop (input, update, render)
- Developed `random_ai()` for random move selection
- Developed `minimax_ai()` for optimal play
- Created a function to simulate AI vs AI battles
- Documented the software design
- Created and tested all features in Python

What we found out:**Screenshot 1: Game Startup & State Generation**A screenshot of a terminal window with a dark background. The terminal shows the following text:

```
sk\ 2\ -\ Graph,\ Search\ \&\ Rules
marcogiacoppo@Marcos-MacBook-Pro Assignment 2 Task 2 - Graph, Search & Rules
es % python3 main.py
5478 game states generated.
```

Below the text, a 3x3 tic-tac-toe board is displayed using ASCII art:

```
  |  |
--|--
  |  |
--|--
  |  |
```

At the bottom of the terminal, it says:

```
The current player is: X
Player 1 (X) - Enter your move (1-9):
```

This demonstrates the successful use of graph-based state generation via adjacency list.

Screenshot 2: Mid-Game Interaction

```
The current player is: X
Player 1 (X) – Enter your move (1-9): 5

  |  |
  |  |
  |  |
  |  |
  |  |
  |  |
  |  |
  |  |
  |  |

The current player is: 0
AI (0) chooses position 1

0 |  |
  |  |
  |  |
  |  |
  |  |
  |  |
  |  |
  |  |
  |  |

The current player is: X
Player 1 (X) – Enter your move (1-9): 2

0 | X |
  |  |
  |  |
  |  |
  |  |
  |  |
  |  |
  |  |
  |  |

The current player is: 0
AI (0) chooses position 8

0 | X |
  |  |
  |  |
  |  |
  |  |
  |  |
  |  |
  |  |
  |  |

The current player is: X
Player 1 (X) – Enter your move (1-9): █
```

Highlights turn-based logic and AI decision-making using Minimax to choose the best spot.

Open issues/risks [Optional – **remove** heading/section if not used!]:

- Minimax may be slow on larger boards
- No input validation for non-integer human input